The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

I also certify that the attached copy of the request for grant of a Patent (Form 1/77) bears an amendment, effected by this office, following a request by the applicant and agreed to by the Comptroller-General.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

[redacted]istration under the Companies Act does not constitute a new legal entity but merely [redacted]ts the company to certain additional company law rules.

Signed

Dated 21 October 2003

Patents Form 1/77

Patents Act 1977
(Rule 16)

THE PATENT OFFICE
B
11 APR 2002
NEWPORT

The Patent Office

11APR02 E710226-1 002761
P01/7700 0.00-0208329.3

1/77

Request for grant of a patent

(See the notes on the back of this form. You can also get and explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office
Cardiff Road
NEWPORT
Gwent NP9 1RH

1. Your reference — **P0982.GBA**

2. Patent application number — **0208329.3**
(The i

11 APR 2002

3. Full name, address and postcode of the or of each applicant. *(underline all surnames)* — **The University of York**

7245376001

Patents ADP number *(if you know it)*

If the applicant is a corporate body, give the country/state of its incorporation — **UNITED KINGDOM**

4. Title of the invention — **DATA PROCESSING, PARTICULARLY IN COMMUNICATION SYSTEMS**

5. Name of your agent *(if you have one)* — MARKS & CLERK ~~LOVEN & CO~~

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)* — SUSSEX HOUSE, 83-85 MOSLEY STREET, MANCHESTER, M2 3LG. ~~Quantum House, 30 Tentercroft Street, LINCOLN, LN5 7DB~~

F51/77
16/10/03

Patents ADP number *(if you know it)* — 18001L ~~4467460003~~

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
(a) any applicant named in part 3 is not an inventor, or
(b) there is an inventor who is not named as an applicant, or
(c) any named applicant is a corporate body
See note (d)) — **Yes**

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	38
Claim(s)	8
Abstract	0
Drawing(s)	16 + 16

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature Loven & Co Date **10 April 2002**

12. Name and daytime telephone number of person to contact in the United Kingdom **K J Loven (01522 801111)**

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or have any questions, please contact the Patent Office on 08459 500505
b) Write your answers in capital letters using black ink or you may type them.
c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.
d) If you have answered 'Yes', Patents Form 7/77 will need to be filed.
e) Once you have filled in this form you must remember to sign and date it.
f) For details of the fee and ways to pay, please contact the Patent Office.

Data Processing, Particularly in Communication Systems

Field of the Invention

This invention relates to methods for solving systems of linear equations and, in particular to increasing speed with which such solutions can be obtained by using a computer system. This invention also relates to a computer system for solving systems of linear equations, the computer system without multipliers and dividers. This invention also relates to applications of the method and computer system in communications and signal processing, specifically to the applications which require solving systems of linear equations arising from the linear least squares (LS) problem. More specifically, this invention also relates to multiuser receivers and adaptive filtering.

Background of the Invention

Linear equations occur frequently in all branches of sciences and engineering, and effective methods are needed for solving them. Furthermore, many science and engineering problems lead not to a single equation, but to a system of equations. Classical methods of solving these systems can be divided into two categories: direct methods and iterative methods.

Direct methods attempt to produce an exact solution by using a finite number of operations. A problem with direct methods, however, is that the number of operations required is large, which makes the methods slow and, also, sensitive to truncation errors. Furthermore, direct methods often fail on ill-conditioned matrices. Also, due to truncation errors and complex operations required such as division and multiplication, direct methods are not well suited for implementation on fixed-point computer systems. Instead, direct methods are implemented preferably on floating-point computer systems. However, floating-point computer systems are slower, more complicated, and more expensive for implementation compared to fixed-point computer systems.

Iterative methods solve a system of equations by repeated refinements of an initial approximation until the result is acceptably close to the solution. Each of the iterations is based on the result of the previous one, and in theory, is supposed to improve the previous approximation. Generally, iterative methods produce an

approximate solution of desired accuracy by yielding a sequence of solutions, which converges to the exact solution as the number of iterations tends to infinity.

In solving systems of equations, especially when the number of equations is large, it is desirable to use a computer system. The word length of the system has a direct bearing on accuracy, and the likelihood of truncation errors increases with the number of operations required for a solution. For this reason, iterative methods are often preferred over direct methods because a solution can be arrived at with fewer operations. Yet, existing iterative methods do not adequately minimise the number of operations required to reach a solution. Another disadvantage of existing iterative methods is that they are not able to avoid truncation errors even if the number of operations is small.

When solving a system of linear equations with a computer system, another consideration is hardware efficiency. For example, it is often desirable to implement iterative methods in hardware, such as FPGA (Field-Programmable Gate Array) or ASIC (Application-Specific Integrated Circuit) platforms. One way to improve efficiency is to exclude operations, which are complex for hardware implementation. One skilled in the field knows that multipliers and dividers occupy areas on FPGA or ASIC, which are one order greater than that of adders. Existing iterative methods exploit multiplication and/or division operations. Therefore, existing iterative methods result in non-efficient hardware implementation.

Another way to improve efficiency is the use of a computer system with fixed-point operations. However, existing iterative methods, when implemented in such systems, lead to low accuracy of the solution. Furthermore, truncation errors can lead to divergence of the existing iterative methods.

Another shortcoming of existing iterative methods is that the number of iterations is not decreased when the solution is sparse; here, the sparse solution means that only a relatively small number of elements of the solution vector with respect to the vector length are not zero or have high amplitudes. Many sciences and engineering problems deal with sparse solutions. For example, in communications when an impulse response of a channel with multipath propagation is estimated, the number of valuable paths can be as small as between 1 and 6, while the number of resolvable delays, defining the size of a system of linear equations, can be as large as several hundreds.

In communications and signal processing, there are many applications where solving systems of linear equations is required. This includes linear equalisation, for

example, zero-forcing and MMSE (minimum mean squares error) equalisation. This also includes measurement of an impulse response of a propagation channel, for example, a multipath channel, a room reverberation, and others. This also includes linear multiuser detection, for example, decorrelating and MMSE multiuser detection. This also includes speech coding, for example, linear predictive coding (LPC). In these and many other applications a system of linear equations may arise from the least squares (LS) problem.

CDMA (Code Division Multiple Access) is a multiple access method based on a spread spectrum technique. CDMA has many advantages over earlier developed FDMA (Frequency Division Multiple Access) and TDMA (Time Division Multiple Access) methods, such as simplicity of frequency planning and spectrum efficiency. In a CDMA method, a narrow-band data signal of a user is multiplied to a relatively broad band by a spreading code having a much broader band than the data signal. All users transmit simultaneously on the same frequency band. On each connection between a base station and a mobile user, a different spreading code is used, and the signals of the users can be distinguished from one another in the receivers on the basis of the spreading codes of the users.

The object of the receiver is to detect the signal of the desired user among a plural number of interfering signals. In the receiver, the data signal is restored to the original band by multiplying the received signal, in correlators or matched filters, by the same spreading code as was used in the transmission step. Ideally, the signals of the other users are not restored to the narrow band. In practice, the spreading codes may partly correlate, and in such instances the signals of the other users make it more difficult to detect the desired signal, by distorting the received signal. This interference caused by the users to one another is called multiuser interference.

Multiuser receivers are effective means for cancelling multiuser interference and such cancelling is known to improve spectral efficiency of multi-user communications networks.

One receiver solution operating in an environment with multiuser interference is a so-called decorrelating receiver, which cancels multiuser interference by multiplying matched-filter outputs by the inverse matrix of the cross-correlation matrix of the spreading codes. A decorrelating receiver is discussed in greater details in R. Lupas, S. Verdu: "Linear multiuser detectors for synchronous code-division multiple access channels", IEEE Transactions on Information Theory, vol. IT-35,

p.123-136, January 1989. In a channel without multipath propagation the cross-correlation matrix can be precomputed and inverted, and the inverse matrix can be stored in the decorrelating receiver; this enables the receiver to avoid calculating the inverse matrix and, thus, simplify the receiver. However, in known decorrelating receivers, multiplying by the inverse matrix is complicated because it requires multiplication operations. Another disadvantage of the decorrelating receiver is that in channels with multipath propagation, i.e., in more realistic channels, the cross-correlation matrix depends on multipath characteristics, therefore the inverse matrix cannot be precomputed and matrix inverse operation is required which is very complicated for real-time implementation, especially for a large number of users. Another disadvantage of known decorrelating receivers is the effect of noise enhancement leading to increasing detection errors.

One skilled in the field knows that a better detection performance is achieved in a so-called MMSE (Minimum Mean Squares Error) multiuser receiver. A disadvantage of the MMSE receiver is that it uses a modified cross-correlation matrix which depends on the noise power and whose inversion cannot be precomputed. Another disadvantage is that, in known MMSE receivers, multiplying by the inverse matrix is complicated because it requires multiplication operations.

Many multiuser receivers (such, for example, as decorrelating receiver, MMSE receiver, and interference cancellation receivers) are implemented on the basis of solving linear systems of equations; this is known, for example, from the paper by Peng H. Tan and Lars K. Rasmussen, "Linear interference cancellation in CDMA based on iterative techniques for linear equation systems", published in the journal IEEE Transactions on Communications, vol. 48, No. 12, Dec. 2000, pages 2099-2108. In multiuser receivers, direct methods for solving systems of linear equations are complicated for implementation. Existing iterative techniques, such as, for example, methods with the steepest descent iteration, the conjugate gradient iteration, the Jacobi iteration, the Gauss-Seidel iteration and others are yet complicated for implementation. Therefore, multiuser receivers are still limited in use, which does not improve spectral efficiency of multi-user communications network, for example, the third generation mobile communication system UMTS, and the like. When used in equipment of a mobile user, for example, in UMTS downlink, multiuser receivers complicated for implementation result in a high power consumption, which shortens the battery life of the equipment.

Adaptive digital filtering is used in many areas of signal processing and communications. Optimal adaptive filtering is based on solving systems of linear equations; for example, this is known from the paper by John F. Doherty, "Channel equalization as a regularized inverse problem" published in the book "The digital signal processing handbook", CRC Press, 1998, page 31-3. Existing direct methods for solving systems of linear equations are complicated for implementation. In practice, iterative methods such as recursive least squares (RLS) and least mean square (LMS) methods are most suited for real-time implementation. The RLS method possesses a fast convergence, however, it is computationally unstable and has a high complexity; in particular, it requires multiplication and division operations. The LMS method can be implemented with no division, but it still requires multiplications. Another disadvantage of the LMS method is that it possesses a low convergence speed, which significantly limits its applications.

Since existing iterative methods exploit multiplication operations they are complicated for hardware implementation, this limits a sampling frequency at which adaptive digital filters can operate and applications where such filters can be used. For example, it is a practical problem of implementing adaptive filters operating at sampling frequencies as high as 100 MHz and higher. Another disadvantage of existing adaptive filters based on iterative methods is that elements of the solution vector are truncated before using them as filter coefficients; this results in a truncation error of adaptive filtering. Another disadvantage of the RLS and LMS methods is that after the convergence there is a final error with respect to an accurate solution; the error is determined, in particular by a constant step-size parameter.

Summary of the Invention

One aspect of the invention is a method for solving systems of linear equations. Elements of a solution vector are represented as fixed-point binary words. The method comprises the steps of initialising a solution vector and an auxiliary vector and bit-wise iterations performing passes through elements of the solution vector, updating elements of the solution and auxiliary vectors in the passes, and repeating the passes until a required accuracy of the solution is achieved. The method solves a system of linear equations by starting the bit-wise iterations with the most significant bit of the words and proceeds to a next (less significant) bit when the solution

achieved for the current bit is not changed in the last pass. The solution is obtained with a computer system programmed to perform a sequence of operations.

Another aspect of the invention is a computer system for solving systems of linear equations, the computer system performing no multiplication or division operations. The computer system comprises: a host processor producing itself or receiving from other devices parameter signals representing elements of a coefficient matrix and right-side vector of the system of linear equations and transmitting to other devices parameter signals representing elements of a solution vector; a host bus coupled to the host processor; an internal bus; a first means for updating and storing elements of the solution vector, the first means coupled to the host and internal buses; a second means for storing elements of the coefficient matrix, the second means coupled to the host and internal buses; a third means for determining successful iterations and preferable updates, the third means coupled to the internal bus and the second means; a fourth means for updating and storing elements of an auxiliary vector, the fourth means coupled to the host bus, internal bus, and the second means.

Another aspect of the invention is a multiuser receiving method in a data transmission system in which code division multiple access involving multiuser interference among respective signals, each signal representing a succession of data signals translated into bits and transmitted at a rate of a plurality of chips per bit, spread by a respective spreading code, is applied for detecting a particular data signal, from among a plurality of data signals, said method comprising: filtering matched with the spreading codes and applied to the received signal to obtain respective output signals; transforming the matched-filter output signals by solving a system of linear equations of the kind $\mathbf{Rh}=\beta$ where $\mathbf{R}$ is a NxN cross-correlation matrix of the spreading codes, β is a $Nx1$ vector grouping the matched-filter output signals, $\mathbf{h}$ is the $Nx1$ solution vector representing the transformed signals, and N is a number of used spreading codes, and subjecting the transformed signals to obtain an estimate of the data signal; wherein solving of the system of linear equations comprises the steps of: representing elements of the solution vector as fixed-point binary words each consisting of at least one bit; initialising the solution vector and an auxiliary vector; performing, for each bit representing the binary words, bit-wise iterations comprising the steps of: performing passes through all elements of the solution vector; updating elements of the solution vector in the passes; updating elements of the auxiliary vector

in the passes; repeating the passes until a finishing condition is fulfilled; and performing the bit-wise iterations until a stopping condition is fulfilled.

Another aspect of the invention is a multiuser receiver performing matched filtering of a received signal with all user-specific spreading codes and transforming result of the matched filtering to a solution vector whose values are represented by a solution of a system of linear equations; the multiuser receiver comprising means to solve the system of linear equations without performing multiplication or division operations comprising: means for matched filtering, a computer system for generating a solution vector **h** in an equation of the kind $\mathbf{Rh}=\beta$, where **R** is the *NxN* cross-correlation matrix of spreading codes, β is the *Nx1* matched-filter vector, and **h** is the *Nx1* solution vector whose elements are to be calculated, and means for estimating data signals from the output signals of the computer system.

Another aspect of the invention provides an adaptive filter having a transfer function determined by a plurality of coefficients whose values are represented by a solution of a system of linear equations; the adaptive filter comprising means to solve the system of linear equations without performing multiplication or division operations, comprising: means for calculating autocorrelation and cross-correlation coefficients, means for generating a solution vector **h** in an equation of the kind $\mathbf{Rh}=\beta$, where **R** is the *NxN* coefficient autocorrelation matrix, β is the *Nx1* cross-correlation vector, and **h** is the *Nx1* solution vector whose elements are to be calculated.

A technical advantage of the invention is that a system of linear equations may be solved with a minimum of operations. A further advantage of the invention is that solution may be obtained without multiplication and division operations, enabling simple hardware implementation. A further advantage of the invention is that solution may be obtained with higher accuracy. Another advantage of the invention when applied to multiuser detection is that spectral efficiency of communication networks is increased and power consumption of multiuser receivers is reduced. Another advantage of the invention when applied to adaptive filtering is that the convergence speed increases and truncation errors are eliminated.

In practical terms, the avoidance of multiplication steps in the algorithm translates into there being no necessity for multipliers in a circuit or chip designed to implement the algorithm. When it is appreciated that a multiplier demands a

complexity that is one order of magnitude greater than an adder, it is immediately evident that substantial savings can be made by the use of the algorithm. This leads to consequential benefits in terms of speed, compactness, reduced power consumption and improved portability of devices incorporating the invention

Brief description of the Drawings

The invention will now be described with reference to the following drawings, in which:

Figure 1 is a flow diagram implementing the method for real-valued data;

Figure 2 is a flow diagram implementing the method for real-valued data and an auxiliary delta array;

Figure 3 is a flow diagram implementing the method for real-valued data, an auxiliary delta array, and a check for the solution amplitude range;

Figure 4 is a flow diagram showing application of the method for complex-valued data;

Figure 5 is a flow chart explaining the steps in searching for the minimum;

Figure 6 is a flow diagram showing application of the method for complex-valued data and an auxiliary delta array;

Figure 7 is an example of a data processing system implementing the method;

Figure 8 is a general scheme of the computer system according to the invention;

Figure 9 represents an **R**-memory block;

Figure 10 represents an **h**-updates block;

Figure 11 represents a **Q**-updates block;

Figure 12 represents a minimisation block;

Figure 13 represents a multiuser receiver;

Figure 14 represents a detector of a multiuser receiver;

Figure 15 represents an adaptive filter;

Figure 16 represents a block for calculating filter coefficients in an adaptive filter.

Detailed Description of the Invention

The invention describes a method for solving systems of linear equations, a computer system for solving systems of linear equations without multiplication or

division operations, a multiuser receiving method, a multiuser receiver, and an adaptive filter.

Prior to description of the preferred embodiments, a description of the least squares problem and coordinate descent optimisation will be made so as to help better understanding of the present invention.

The linear least squares (LS) problem deals with minimisation of the function

$$J(\mathbf{h}) = |\mathbf{Zh} - \mathbf{d}|^2$$

with respect to an unknown *Nx1* vector **h**, where **Z** is an *MxN* matrix, and **d** is a *Mx1* vector. This problem is known to be equivalent to solving a system of linear equations (normal equations):

$$\mathbf{Rh} = \boldsymbol{\beta}$$

where **R** is an *NxN* coefficient matrix

$$\mathbf{R} = \mathbf{Z}^T\mathbf{Z}$$

β is an *Nx1* right-side vector

$$\boldsymbol{\beta} = \mathbf{Z}^T\mathbf{d}$$

and $()^T$ denotes matrix transpose. The function $J(\mathbf{h})$ can be written as

$$J(\mathbf{h}) = \frac{1}{2}\sum_{m=1}^{N}\sum_{n=1}^{N} R(m,n)h(m)h(n) - \sum_{m=1}^{N}\beta(m)h(m)$$

where $h(m)$ is m^{th} element of the vector **h**, $\beta(m)$ is m^{th} element of the vector β, and $R(m,n)$ is $(m,n)^{th}$ element of the matrix **R**.

In practical circumstances, elements of the solution vector **h** are limited in amplitude. Therefore, minimising the function $J(\mathbf{h})$ with respect to an *N*-dimensional vector **h** is considered under constraints

$$\mathbf{h} \in \mathbf{U} = \{[h(1),...,h(m),...,h(N)] : |h(m)| \le H, m = 1,...,N\}$$

where $H>0$ is a known number.

A number of computerised techniques are known for minimising a function $J(\mathbf{h})$; coordinate descent optimisation is one of them. The coordinate descent optimisation can be described as follows. Let $\mathbf{e}_i$ = (0,...,1,...,0) be a unity basis vector, whose i^{th} coordinate is 1 and the others are 0. Let $\mathbf{h}_0$ be an initial value of the vector **h** and α_0 be a positive number. Let us also assume that we know a vector $\mathbf{h}_k$ and a number α_k for some $k \ge 0$. Denote

$$\mathbf{p}_k = \mathbf{e}_{i_k}, \quad i_k = k - N\left[\frac{k}{N}\right] + 1$$

where *[k/N]* denotes the integer part of the number *k/N*; this means that

$$\mathbf{p}_1 = \mathbf{e}_1, \mathbf{p}_2 = \mathbf{e}_2, ..., \mathbf{p}_N = \mathbf{e}_N,$$
$$\mathbf{p}_{N+1} = \mathbf{e}_1, \mathbf{p}_{N+2} = \mathbf{e}_2, ..., \mathbf{p}_{2N} = \mathbf{e}_N,$$
$$.....$$

Let us calculate the function $J(\mathbf{h})$ at the point $\mathbf{h}=\mathbf{h}_k+\alpha_k\mathbf{p}_k$ and check the condition

$$\mathbf{h}_k + \alpha_k \mathbf{p}_k \in \mathbf{U}, \quad J(\mathbf{h}_k + \alpha_k \mathbf{p}_k) < J(\mathbf{h}_k) \qquad (1)$$

If this condition is fulfilled, then

$$\mathbf{h}_{k+1} = \mathbf{h}_k + \alpha_k \mathbf{p}_k, \quad \alpha_{k+1} = \alpha_k$$

If this condition is not fulfilled we calculate the function $J(\mathbf{h})$ at the point $\mathbf{h}=\mathbf{h}_k-\alpha_k\mathbf{p}_k$ and check the condition

$$\mathbf{h}_k - \alpha_k \mathbf{p}_k \in \mathbf{U}, \quad J(\mathbf{h}_k - \alpha_k \mathbf{p}_k) < J(\mathbf{h}_k) \qquad (2)$$

If this new condition is fulfilled, then

$$\mathbf{h}_{k+1} = \mathbf{h}_k - \alpha_k \mathbf{p}_k, \quad \alpha_{k+1} = \alpha_k$$

Let us call a $(k+1)^{th}$ iteration successful, if either condition (1) or (2) is fulfilled. If a $(k+1)^{th}$ iteration is not successful, then

$$\mathbf{h}_{k+1} = \mathbf{h}_k, \quad \alpha_{k+1} = \begin{cases} \lambda\alpha_k, \text{ if } i_k = N, \text{ and } \quad \mathbf{h}_k = \mathbf{h}_{k-N+1} \\ \alpha_k, \text{ otherwise} \end{cases} \qquad (3)$$

Here λ $(0<\lambda<1)$ is a parameter. The condition in (3) means that if in last N iterations with a step size parameter α_k there is at least one successful iteration, then the step size parameter α_k is kept for the following N iterations. If among the previous N iterations there is no successful one, then the step size parameter α_k is reduced.

In the described coordinate descent optimisation, if the function $J(\mathbf{h})$ is convex and differentiable on $\mathbf{U}$, then the sequence $\mathbf{h}_k$ converges to the solution $\mathbf{h}$ for any initial values $\mathbf{h}_0$ in $\mathbf{U}$, $\alpha_0>0$, and $0<\lambda<1$. This result can be found, for example, in the book by F. P. Vasiliev, "Numerical methods for solving optimisation problems" (in Russian), Nauka, Moscow, 1988, page 345.

In the LS problem the function $J(\mathbf{h})$ is convex and differentiable; this is known, for example, from the book by F. P. Vasiliev, "Numerical methods for solving optimisation problems" (in Russian), Nauka, Moscow, 1988, page 169. Therefore, the coordinate descent optimisation when applied to the LS problem provides convergence to the solution. The coordinate descent optimisation applied for finding the minimum of the function $J(\mathbf{h})$ in the LS problem is as follows.

Let us $\mathbf{p}_k=\mathbf{e}_i$. At the $(k+1)^{th}$ iteration the conditions

$$\Delta J(\mathbf{h}_k) = J(\mathbf{h}_k + \alpha_k \mathbf{e}_i) - J(\mathbf{h}_k) < 0$$

and

$$\Delta J(\mathbf{h}_k) = J(\mathbf{h}_k - \alpha_k \mathbf{e}_i) - J(\mathbf{h}_k) < 0$$

should be checked. Since only i^{th} element in the vector $\mathbf{e}_i$ is not zero and the matrix $\mathbf{R}$ is symmetric, meaning that $R(m,n)=R(n,m)$, we have

$$\Delta J(\mathbf{h}_k) = J(\mathbf{h}_k + \alpha_k \mathbf{e}_i) - J(\mathbf{h}_k) = \frac{\alpha_k}{2}\left[Q(i) + \alpha_k R(i,i)\right]$$

where

$$Q(i) = -2\beta(i) + 2\sum_{m=1}^{N} h^{(k)}(m) R(m,i)$$

and $h^{(k)}(m)$ is m^{th} element of the vector $\mathbf{h}_k$. As $\alpha_k>0$ the condition

$$\Delta J(\mathbf{h}_k) = J(\mathbf{h}_k + \alpha_k \mathbf{e}_{i_k}) - J(\mathbf{h}_k) < 0$$

can now be represented as

$$\alpha_k R(i,i) < -Q(i)$$

Similarly, the condition

$$\Delta J(\mathbf{h}_k) = J(\mathbf{h}_k - \alpha_k \mathbf{e}_{i_k}) - J(\mathbf{h}_k) < 0$$

can be represented as

$$\alpha_k R(i,i) > Q(i)$$

Let $\mathbf{Q}_k$ denote an auxiliary vector $\mathbf{Q}$ at k^{th} iteration. If the $(k+1)^{th}$ iteration is not successful, then

$$\mathbf{h}_{k+1} = \mathbf{h}_k,$$
$$\mathbf{Q}_{k+1} = \mathbf{Q}_k$$

If the iteration is successful, then

$$h^{(k+1)}(i) = h^{(k)}(i) \pm \alpha_k,$$
$$Q^{(k+1)}(n) = Q^{(k)}(n) \pm 2\alpha_k R(n,i), \quad n = 1,\ldots,N.$$

where $Q^{(k)}(n)$ denotes n^{th} element of the vector $\mathbf{Q}_k$. Initial values $\mathbf{h}_0$ and $\mathbf{Q}_0$ of the vectors $\mathbf{h}$ and $\mathbf{Q}$ can be chosen, for example, as follows:

$$h^{(0)}(n) = 0, \quad n = 1,\ldots,N,$$
$$Q^{(0)}(n) = -2\beta(n), \quad n = 1,\ldots,N.$$

This means that the vector $\mathbf{h}$ is initialised by zeros, and the vector $\mathbf{Q}$ is initialised

by the right side vector β of the system of linear equations $\mathbf{Rh}=\beta$.

If we choose $H=2^q$ where q is an integer, $\alpha_0=H/2$, and $\lambda=1/2$, we avoid explicit multiplications by the step-size parameter α_k. Such a choice allows multiplications to be implemented by means of bit-shift operations. Also, the described technique does not use any division operations.

The method according to the invention may be characterised by the following parameters: (1) $\mathbf{R}$ is an NxN coefficient matrix of the system of linear equations; (2) β is a $Nx1$ right-side vector of the system of linear equations; (2) $\mathbf{h}$ is a $Nx1$ solution vector of the system of linear equations; (1) N is the size of the solution vector; (2) M is the number of bits representing the solution; (3) Nit is the maximum number of passes for each bit of the solution; (4) H is a positive number defining the amplitude range of the solution.

For example, the method of solving systems of linear equations in accordance with the invention can be described as follows.

Elements of the solution vector are represented as fixed-point binary words each consisting of at least one bit. An initialisation of the solution vector and an auxiliary vector are performed. The method performs bit-wise iterations for solving the system of linear equations, starting from the most significant bit and proceeding with the next less significant bit if a finishing condition is fulfilled. For each bit representing the binary words, the method performs at least one pass through all elements of the solution vector, updating elements of the solution vector. The method stops solving the system of linear equations when a stopping condition is fulfilled.

Elements of the solution vector are initialised as zeros; other initialisations are also possible. Elements of the auxiliary vector are initialised by corresponding elements of the right-side vector of the system of linear equations; other initialisations are also possible.

In the pass, in turn for each element of the solution vector a condition successful/unsuccessful is checked. If successful, the solution vector and the auxiliary vectors are updated.

In the solution vector, the only element is updated, for which the condition successful/unsuccessful is checked.

When checking the condition successful/unsuccessful for an element of the solution vector, a finite number of possible updates of the element are checked to find a preferable update. The possible number of updates can be, for example, two (for real-valued data) or four (for complex-valued data); other variants are also possible. For each possible update an auxiliary value is calculated based on the corresponding element of the auxiliary vector and the minimum among the auxiliary values is found. The minimum is compared with a threshold calculated by using a diagonal element of the coefficient matrix, the diagonal element corresponding to the checked element of the solution vector, and a step-size parameter. The condition is successful if the minimum is less than the threshold, and it is unsuccessful if the minimum is higher than or equal to the threshold. The minimum indicates the preferable update and the updated element of the solution vector is replaced by the preferable update. For the successful condition, the auxiliary vector is updated by using elements of a row of the coefficient matrix; this row corresponds to the updated element of the solution vector.

When analysing an element of the solution vector the order of the analysis can be either arbitrary or a specific, for example, starting from an element whose position corresponds to the position of an element of the auxiliary vector with a maximum amplitude and in the order of reducing the amplitude. In such a case, the method may also include an operation of ordering the elements of the auxiliary vector.

The finishing condition is that in the last pass there was no successful update.

The step-size parameter is decreased after analysing each bit of the solution; preferably, this decrease is by a factor of two.

The method stops when a predefined number M of bit-wise iterations has been performed; this number defines valuable bits of the binary words representing elements of the solution vector. The method can stop if a condition that a predefined number *Nit* of passes through all elements of the solution vector is exceeded in a bit-wise iteration. Another condition to stop the method may be that the computer time intended for performing this method is finished. Other conditions may also be used.

The solution can be obtained, for example, with a computer system programmed to perform a sequence of operations.

1. Implementation of the method for real-valued data

For real-valued data the method can be implemented by using the following sequence

of operations as represented in Figure 1.

1) Initialisation of the solution vector **h** and an auxiliary vector **Q**: $h(r)=0$, $Q(r)=-\beta(r)$, $r=1, \ldots, N$. Also, at this step, a bit number m, a pass number it, and an element number p are set to zero: $m=0$, $it=0$, $p=0$.

2) Bit number is increased: $m=m+1$.

3) A step-size parameter is calculated as $d=2^{-m}H$. The preferable way to chose the amplitude range of the solution is $H=2^q$ where q is an integer so that the step-size parameter is $d=2^{q-m}$ and further operations involving d are performed without multiplications.

4) Pass number is increased: $it=it+1$. A binary parameter *Flag* is set to zero: $Flag=0$.

5) Element number p is increased: $p=p+1$.

6) An argument is found which indicates the minimum of the three values: $Q(p)$, $-Q(p)$, and $-(d/2)R(p,p)$: $arg=min\{Q(p), -Q(p), -(d/2)R(p,p)\}$. For calculating the third value, we avoid explicit multiplication because the step-size parameter d is a power of two and this multiplication is performed by a bit-shift operation. For finding the minimum we need to perform two operations of comparison, which require no multiplication.

7) The condition $arg=1,2, or\ 3$ is checked.

8) If $arg=1$ then the following update of the element of the solution vector is performed: $h(p)=h(p)+d$. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+d\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that this iteration is successful. If $arg=2$ then the following update of the solution is performed: $h(p)=h(p)-d$. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)-d\cdot R(p,r)$ for $r=1,\ldots,N$; the

binary parameter *Flag* is set to one: *Flag=1*, indicating that this iteration is successful.

9) The condition $p=N$ is checked. This equality means that in the current pass all N elements of the solution vector have been already analysed. If this condition is not satisfied, the algorithm analyses the next element (*step 5*).

10) If the condition $p=N$ is satisfied the algorithm checks another condition: *Flag=0*.

11) If the latter is not true, i.e., *Flag=1* meaning that in the course of the last pass, including sequential analysis of all N elements of the solution vector, there is at least one successful update of elements of the solution vector **h** and the auxiliary vector **Q**, then a stopping condition *"iterations?"* is checked. This condition can be, for example, $it=Nit$, i.e., that the pass number *it* is equal to the maximum possible number of passes *Nit* which is a predefined parameter. This condition can also be based on checking the rest of a computer time predefined for solving the system of linear equations; other conditions can also be used. If this condition is satisfied then the method stops; in such a case the vector **h** contains N elements of the current solution. The bit number m characterises the accuracy of the solution.

12) If the condition *"iterations?"* is not satisfied, a new pass is performed (*step 4*).

13) If *Flag=0*, the condition $m=M$ is checked; this condition means that all M bits of the solution have been already analysed. If the condition $m=M$ is not satisfied, the method proceeds with analysing the next bit of the solution (*step 2*), i.e., performs a new bit-wise iteration.

14) If $m=M$, the method stops; in such a case the vector **h** contains N elements of the solution. The predefined number of bits M characterises a predefined

accuracy of the solution.

2. Another implementation of the method for real-valued data

For real-valued data the method can also be implemented by using an auxiliary delta array Δ and the following operations as represented in Figure 2.

1) Initialisation of the solution vector **h** and an auxiliary vector **Q**: $h(r)=0$, $Q(r)=-\beta(r)$, $r=1, \ldots, N$. A bit number m, a pass number it, and an element number p are set to zero: $m=0$, $it=0$, $p=0$.

2) Bit number is increased: $m=m+1$.

3) A step-size parameter is calculated as $d=2^{-m}H$. The preferable way to chose the amplitude range of the solution is $H=2^q$ where q is an integer so that the step-size parameter is $d=2^{q-m}$ and further operations involving d are performed without multiplications. An delta array is initialised; this array has two elements: $\Delta(1)=d$ and $\Delta(2)=-d$.

4) Pass number is increased: $it=it+1$. A binary parameter *Flag* is set to zero: $Flag=0$.

5) Element number p is increased: $p=p+1$.

6) An argument is found which indicates the minimum of the three values: $Q(p)$, $-Q(p)$, and $-(d/2)R(p,p)$: $arg=min\{Q(p), -Q(p), -(d/2)R(p,p)\}$. For calculating the third value, we avoid explicit multiplication because the step-size parameter d is a power of two and this multiplication is performed by a bit-shift operation. For finding the minimum we need to perform two operations of comparison, which require no multiplication.

7) The condition $arg < 3$ is checked.

8) If $arg<3$ then the following update of the solution is performed: $h(p)=h(p)+\Delta(arg)$. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+\Delta(arg)\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that this iteration is successful.

9) The condition $p=N$ is checked. This equality means that in the current pass all N elements of the solution vector have been already analysed. If this condition is not satisfied, the next element of the solution vector is analysed (*step 5*).

10) If the condition $p=N$ is satisfied, another condition is checked: $Flag=0$.

11) If the latter is not true, i.e., $Flag=1$, meaning that in the course of the last pass, including sequential analysis of all N elements of the solution vector, there is at least one update of elements of the solution vector **h** and the auxiliary vector **Q**, then a stopping condition *"iterations?"* is checked. This condition can be, for example, $it=Nit$, i.e., that the pass number *it* is equal to the maximum possible number of passes *Nit* which is a predefined parameter. This condition can also be based on checking the rest of a computer time intended for solving the system of linear equations; other conditions can also be used. If this condition is satisfied the methods stops; in such a case the vector **h** contains N elements of the solution. The bit number m characterises the accuracy of the solution.

12) If the condition *"iterations?"* is not satisfied, a new pass is performed (*step 4*).

13) If $Flag=0$, the condition $m=M$ is checked; this condition means that all M bits of the solution have been already analysed. If the condition $m=M$ is not satisfied, a new bit-wise iteration is performed by analysing the next bit of the

solution (*step 2*).

14) If $m=M$, the methods stops; in such a case the vector **h** contains N elements of the solution vector with the predefined number M of valuable bits. The predefined number of bits M characterises a predefined accuracy of the solution.

3. Program implementing the method for real-value data

An example of implementation of the method in accordance with the invention for real-valuable data with an auxiliary delta array by a MATLAB program is as follows.

```
function h=LS_real(R,N,M,H,Nit,beta);
h=zeros(N,1);
Q=-beta;
d=H/2;
for m=1:M
   delta(1)=d; delta(2)=-d;
   d=d/2;
   for it=1:Nit
       Flag=0;
       for p=1:N
           [val arg]=min([Q(p),-Q(p),-abs(R(p,p)*d)]);
           if arg<3
              Flag=1;
              h(p)=h(p)+delta(arg);
              Q=Q+delta(arg)*R(p,:);
           end
       end
       if Flag==0 break; end
   end
end
```

4. Another implementation of the method for real-valued data

In some cases it is preferable to obtain such a solution of the system of the equations that the amplitude range for the solution is predefined, i.e., there are lower h_1 and upper h_2 bounds for elements of the solution vector; for example, $h_1=0$ which means that the elements of the solution vector should be positive. For real-valued data the method can be implemented by using the following operations as represented in

Figure 3.

1) Initialisation of the solution vector **h** and an auxiliary vector **Q**: $h(r)=0$, $Q(r)=-\beta(r)$, $r=1, \ldots, N$. A bit number m, a pass number it, and an element number p are set to zero: $m=0$, $it=0$, $p=0$.

2) Bit number is increased: $m=m+1$.

3) A step-size parameter is calculated as $d=2^{-m}H$. The preferable way to chose the amplitude range of the solution is $H=2^q$ where q is an integer so that the step-size parameter is $d=2^{q-m}$ and further operations involving d are performed without multiplications. An delta array is initialised; this array has two elements: $\Delta(1)=d$ and $\Delta(2)=-d$.

4) Pass number is increased: $it=it+1$. A binary parameter *Flag* is set to zero: $Flag=0$.

5) Element number p is increased: $p=p+1$.

6) An argument is found which indicates the minimum of the three values: $Q(p)$, $-Q(p)$, and $-(d/2)R(p,p)$: $arg=min\{Q(p), -Q(p), -(d/2)R(p,p)\}$. For calculating the third value, we avoid explicit multiplication because the step-size parameter d is a power of two and this multiplication is performed by a bit-shift operation. For finding the minimum we need to perform two operations of comparison, which require no multiplication.

7) The condition $arg < 3$ is checked.

8) If $arg<3$ then the following update of the solution is performed: $h(p)=h(p)+\Delta(arg)$. Then the condition $h_1 \le h(p) \le h_2$ is checked. If it is true, then all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+\Delta(arg)\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that this iteration is successful. If the condition $h_1 \le h(p) \le h_2$ is not satisfied then the

following update of the solution is performed: $h(p)=h(p)-\Delta(arg)$. Note that a more general approach can be implemented at this step when the condition $h_1(p) \leq h(p) \leq h_2(p)$ is checked; here, $h_1(p)$ and $h_2(p)$ are elements of two predefined $N \times 1$ vectors containing lower and upper bounds which are individual for each element of the solution vector.

9) The condition $p=N$ is checked. This equality means that in the current pass all N elements of the solution vector have been already analysed. If this condition is not satisfied, the next element is analysed (*step 5*).

10) If the condition $p=N$ is satisfied, then another condition is checked: $Flag=0$.

11) If the latter is not true, i.e., $Flag=1$, meaning that in the course of the last iteration, including sequential analysis of all N elements of the solution, there is at least one successful update of elements of the solution vector **h** and the auxiliary vector **Q**, then a stopping condition *"iterations?"* is checked. This condition can be, for example, $it=Nit$, i.e., that the pass number it is equal to the maximum possible number of passes Nit which is a predefined parameter. This condition can also be based on checking the rest of a computer time intended for solving the system of linear equations; other conditions can also be used. If this condition is satisfied the method stops; in such a case the vector **h** contains N elements of the solution. The bit number m characterises an accuracy of the solution.

12) If the condition *"iterations?"* is not satisfied, a new pass is performed (*step 4*).

13) If $Flag=0$, the condition $m=M$ is checked; this condition means that all M bits of the solution have been already analysed. If the condition $m=M$ is not satisfied, a new bit-wise iteration is performed by analysing the next bit of the solution (*step 2*).

14) If $m=M$, the method stops; in such a case the vector **h** contains N elements of the solution. The predefined number of bits M characterises a predefined accuracy of the solution.

5. A general implementation of the method for complex-valued data

In order to solve a linear system of equations **Rh=β** with complex-valued data one can use implementations of the method for real-valued data (for example, that described above) in application to a new real-valued system **Ab=c** with the following well-known replacements: the size of the real-valued system of linear equations is *2Nx2N* (**A** is a *2Nx2N* real-valued coefficient matrix, **b** is a *2Nx1* real-valued solution vector, and **c** is a *2Nx1* real-valued right-side vector); elements of the vector **b** are defined as $b(2n-1)=Re\{h(n)\}$ and $b(2n)=Im\{h(n)\}$; elements of the vector **c** are $c(2n-1)=Re\{\beta(n)\}$ $c(2n)=Im\{\beta(n)\}$; elements of the matrix **A** are $A(2m-1,2n-1)=Re\{R(m,n)\}$, $A(2m,2n)=Re\{R(m,n)\}$, $A(2m-1,2n)=Im\{R(m,n)\}$, and $A(2m,2n-1)=-Im\{R(m,n)\}$; *Re{}* and *Im{}* denote, respectively, real and imaginary parts of a complex number.

6. Implementation of the method for complex-valued data

For complex-valued data the method can also be implemented by using the following sequence of operations as represented in Figure 4.

1) Initialisation of the solution vector **h** and an auxiliary vector **Q**: $h(r)=0$, $Q(r)=-\beta(r)$, $r=1, \ldots, N$. A bit number m, a pass number it, and an element number p are set to zero: $m=0$, $it=0$, $p=0$.

2) Bit number is increased: $m=m+1$.

3) A step-size parameter is calculated as $d=2^{-m}H$. The preferable way to chose the amplitude range of the solution is $H=2^q$ where q is an integer so that the

step-size parameter is $d=2^{q-m}$ and further operations involving d are performed without multiplications.

4) Pass number is increased: $it=it+1$. A binary parameter *Flag* is set to zero: $Flag=0$.

5) Element number p is increased: $p=p+1$.

6) An argument is found which indicates the minimum of the five values: $Re[Q(p)]$, $-Re[Q(p)]$, $Im[Q(p)]$, $-Im[Q(p)]$, and $-(d/2)R(p,p)$: $arg=min\{Re[Q(p)], -Re[Q(p)], Im[Q(p)], -Im[Q(p)], -(d/2)R(p,p)\}$. For calculating the last value, we avoid explicit multiplication because the step-size parameter d is a power of two and this multiplication is performed by a bit-shift operation. For finding the minimum we need to perform operations of comparison, which require no multiplication.

7) The condition $arg=1,2,3,4$ *or* 5 is checked.

8) If $arg=1$ then the following update of the solution is performed: $h(p)=h(p)+d$; the latter means that the real part of the element $h(p)$ is increased by d. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+d\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that the current pass is successful. If $arg=2$ then the following update of the solution is performed: $h(p)=h(p)-d$; the latter means that the real part of the element $h(p)$ is decreased by d. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)-d\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that the current pass is successful. If $arg=3$ then the following update of the solution is performed: $h(p)=h(p)+i\cdot d$ where $i^2=-1$; the latter means that the imaginary part of the element $h(p)$ is increased by d. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+i\cdot d\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag*

is set to one: *Flag=1*, indicating that the current pass is successful. If *arg=4* then the following update of the solution is performed: $h(p)=h(p)-i\cdot d$; the latter means that the imaginary part of the element *h(p)* is decreased by *d*. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)-i\cdot d\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: *Flag=1*, indicating that the current pass is successful.

9) The condition $p=N$ is checked. This equality means that in the current pass all *N* elements of the solution vector have been already analysed. If this condition is not satisfied, the next element is analysed (*step 5*).

10) If the condition $p=N$ is satisfied, another condition is checked: *Flag=0*.

11) If the latter is not true, i.e., *Flag=1*, meaning that in the course of the last pass, including sequential analysis of all *N* elements of the solution, there is at least one successful update of elements of the solution vector **h** and the auxiliary vector **Q**, then a stopping condition *"iterations?"* is checked. This condition can be, for example, *it=Nit*, i.e., that the pass number *it* is equal to the maximum possible number of passes *Nit* which is a predefined parameter. This condition can also be based on checking the rest of a computer time intended for this algorithm; other conditions can also be used. If the stopping condition is satisfied the method stops; in such a case the vector **h** contains *N* elements of the current solution. The bit number *m* characterises the accuracy of the solution.

12) If the condition *"iterations?"* is not satisfied, a new pass is performed (*step 4*).

13) If *Flag=0*, the condition $m=M$ is checked; this condition means that all *M* bits of the solution have been already analysed. If the condition $m=M$ is not satisfied, the next bit-wise iteration is performed by analysing the next bit of

the solution (*step 2*).

14) If $m=M$, the method stops; in such a case the vector **h** contains N elements of the current solution. The predefined number of bits M characterises an accuracy of the solution.

In step 6 of the implementation described above, finding $arg=min\{Re[Q(p)], -Re[Q(p)], Im[Q(p)], -Im[Q(p)], -(d/2)R(p,p)\}$ is required. This can be implemented by using the following operations as represented in Figure 5.

1) Input data are received: $a=Re[Q(p)]$ and $b=Im[(Q(p)]$.

2) The processing of the input data a and b is performed in parallel as follows. The condition $a>0$ is checked. If this condition is satisfied, then $a=-a$ and a binary parameter $I_a=1$. If the condition $a>0$ is not satisfied, then $I_a=0$. The condition $b>0$ is checked. If this condition is satisfied, then $b=-b$ and a binary parameter $I_b=1$. If the condition $b>0$ is not satisfied, then $I_b=0$.

3) The condition $a>b$ is checked. If it is satisfied, then $I_1=I_b$, $I_2=1$, and $c=b$. If the condition is not satisfied then $I_1=I_a$, $I_2=0$, and $c=a$.

4) The condition $c>-(d/2)\cdot R(p,p)$ is checked. If it is true, then $I_3=1$; if it is false, then $I_3=0$.

5) The binary values I_1, I_2, I_3 indicate the position of the minimum: $I_3=0/1$ means that (an update)/(no update) of the solution is required; $I_2=0/1$ means that the real/imaginary part of the solution should be updated; $I_1=0/1$ means that the update should be positive/negative.

7. Another implementation of the method for complex-valued data

For complex-valued data the method can also be implemented by using an auxiliary delta array Δ and the following operations as represented in Figure 6.

1) Initialisation of the solution vector **h** and an auxiliary vector **Q**: $h(r)=0$, $Q(r)=-\beta(r)$, $r=1, \ldots, N$. A bit number m, a pass number it, and an element number p are set to zero: $m=0$, $it=0$, $p=0$.

2) Bit number is increased: $m=m+1$.

3) A step-size parameter is calculated as $d=2^{-m}H$. The preferable way to chose the amplitude range of the solution is $H=2^q$ where q is an integer so that the step-size parameter is $d=2^{q-m}$ and further operations involving d are performed without multiplications. An delta array is initialised; this array has four elements: $\Delta(1)=d$, $\Delta(2)=-d$, $\Delta(3)=i\cdot d$, and $\Delta(4)=-i\cdot d$.

4) Pass number is increased: $it=it+1$. A binary parameter *Flag* is set to zero: $Flag=0$.

5) Element number p is increased: $p=p+1$.

6) An argument is found which indicates the minimum of the five values: $Re[Q(p])$, $-Re[Q(p)]$, $Im[Q(p])$, $-Im[Q(p)]$, and $-(d/2)R(p,p)$: $arg=min\{Re[Q(p]), -Re[Q(p]), Im[Q(p]), -Im[Q(p)], -(d/2)R(p,p)\}$. For calculating the last value, we avoid explicit multiplication because the step-size parameter d is a power of two and this multiplication is performed by a bit-shift operation. For finding the minimum we need to perform operations of comparison, which require no multiplication.

7) The condition $arg< 5$ is checked.

8) If $arg<5$ then the following update of the solution is performed: $h(p)=h(p)+\Delta(arg)$. Also, all elements of the auxiliary vector **Q** are updated as $Q(r)=Q(r)+\Delta(arg)\cdot R(p,r)$ for $r=1,\ldots,N$; the binary parameter *Flag* is set to one: $Flag=1$, indicating that the current iteration is successful.

9) The condition $p=N$ is checked. This equality means that in the current pass all N elements of the solution vector have been already analysed. If this condition is not satisfied, the next element is analysed (*step 5*).

10) If the condition $p=N$ is satisfied, another condition is checked: $Flag=0$.

11) If the latter is not true, i.e., $Flag=1$, meaning that in the course of the last pass, including sequential analysis of all N elements of the solution, there is at least one successful update of elements of the solution vector **h** and the auxiliary vector **Q**, then a stopping condition *"iterations?"* is checked. This condition can be, for example, $it=Nit$, i.e., that the pass number it is equal to the maximum possible number of passes Nit which is a predefined parameter. This condition can also be based on checking the rest of a computer time intended for this algorithm; other conditions can also be used. If this condition is satisfied the method stops; in such a case the vector **h** contains N elements of the current solution. The bit number m characterises an accuracy of the solution.

12) If the condition *"iterations?"* is not satisfied, a new pass is performed (*step 4*).

13) If $Flag=0$, the condition $m=M$ is checked; this condition means that all M bits of the solution have been already analysed. If the condition $m=M$ is not satisfied, the next bit-wise iteration is performed by analysing the next bit of the solution (*step 2*).

14) If $m=M$, the method stops; in such a case the vector **h** contains N elements of the solution. The predefined number of bits M characterises an accuracy of the solution.

8. Program implementing the method for complex-valued data

An example of implementation of the described method for complex-valued data with an auxiliary delta array by a MATLAB program is as follows.

```
function h=LS_cmplx(R,N,M,H,Nit,beta);
h=zeros(N,1);
Q=-beta;
d=H/2;
for m=1:M
   delta(1)=d; delta(2)=-d; delta(3)=i*d; delta(4)=-i*d;
   d=d/2;
   for it=1:Nit
      Flag=0;
      for p=1:N
         [val arg]=min([real(Q(p)),-real(Q(p)), . . .
                        imag(Q(p)),-imag(Q(p)),-R(p,p)*d]);
          if arg<5
             Flag=1;
             h(p)=h(p)+delta(arg);
             Q=Q+delta(arg)*R(p,:);
          end
      end
      if Flag==0 break; end
   end
end
```

Figure 7 shows a computer system for solving systems of linear equations by the method according to the invention. The computer system comprises a parameter interface and a computer processor. The parameter interface receives parameter signals representing the coefficient matrix **R** and the vector β of the system of linear equations $\mathbf{Rh}=\beta$. The output of the computer processor comprises elements of the solution vector **h**.

9. Computer system

Figure 8 represents a computer system in accordance with the invention, the computer system implementing the method of solving systems of linear equations in accordance with the invention. The computer system comprises a Host processor, a Block of h-updates, a Block of Q-updates, a Block of R-memory, a Block of minimisation, and a Controller; it also contains a Host Bus and an Internal Bus. The

Host processor is connected with the Block of h-updates, the Block of Q-updates, the Block of R-memory, and the Controller by means of the Host Bus. The Controller is connected with the Block of h-updates, the Block of Q-updates, the Block of R-memory, and the Block of minimisation by means of the Internal Bus. In addition, the Block of Q-updates is connected with the Block of minimisation, and the Block of R-memory is connected with the Block of Q-updates and the Block of minimisation.

The Host processor produces itself or receives from other devices (not shown here) parameter signals representing the coefficient matrix **R** and the vector β of the linear system $\mathbf{Rh}=\beta$.

There are three stages of operation of the computer system: an initialisation stage, a calculating stage, and a stage of reading results.

At the initialisation stage, the Host processor first sends (through the Host Bus) to all blocks a signal *Init* meaning the beginning of the initialisation stage, and then the Host processor sends signals representing the vector β to the Block of Q-updates and signals representing the coefficient matrix **R** to the Block of R-memory. In some implementations, the signals representing the coefficient matrix **R** can be stored in the Block of R-memory before the initialisation stage. Also, the Host processor sends signals representing parameters of the method through the Host Bus to the other blocks, preferably to the Controller; these parameters may be the size of the solution vector (N), number of bits representing the solution (M), maximum number of passes (Nit), amplitude range of the solution (H). Also, among these parameters may be a maximum operation time, i.e., a time duration during which the calculation of the solution should be done. In some implementations, these parameters can be stored in the Controller instead of sending them by the Host processor. Upon receiving the signal *Init* the Block of h-updates sets elements of the vector **h** to zero.

At the calculating stage, the Controller forms control signals needed for performing the method and sends them to the other blocks through the Internal Bus and to the Host processor through the Host Bus. The control signals may comprise the

bit number m, the pass number it, the element number p, a number r of the element to be updated, signals representing commands to begin and stop operations, address signals to memory blocks; other control signals can also be sent by the Controller to the blocks. The Block of h-updates updates the solution vector **h**. The Block of Q-updates updates the auxiliary vector **Q**. The Block of R-memory sends signals representing the coefficient matrix **R** to the Block of Q-updates and the Block of minimisation according to address signals received from the Controller. The Block of minimisation receives (from the Block of Q-updates) signals representing elements of the auxiliary vector **Q** and (from the Block of R-memory) elements of the matrix **R** and finds a position (*arg*) of the minimum required for controlling the updates of the vectors **h** and **Q**. Signals representing the parameter *arg* are sent to the Internal Bus, from where they are received by the Controller and may also be received by the other blocks.

At the stage of reading results the Host processor forms address signals for the solution vector **h**, sends these signals to the Block of h-updates and reads signals representing the solution vector **h**.

In some implementations, the Host processor can stop solving the linear system at any moment and read the current solution from the Block of h-updates. It can also read other parameters, such, for example, as the current bit number m. The latter allows the Host processor to obtain information of the accuracy of the current solution.

In some implementations, functions of the Controller may be performed by the Host processor; in such a case the Host Bus and the Internal Bus may be connected. In some implementations, the Block of minimisation may be included in the Block of Q-updates. Other reconfigurations of the computer system are also possible.

A variant of implementation of the block of R-memory is shown in Figure 9.

At the initialisation stage, the block receives (from the Host Bus) the signal *Init*, indicating the initialisation stage, and signals representing row (*Init row address*) and

column (*Init column address*) numbers of elements of the coefficient matrix **R** as well as elements of the matrix (*Coefficients of the matrix* **R**). The signal *Init* at the address inputs of the Row and Column multiplexers allows the multiplexers to transmit the address signals to the address input of a Memory block of the coefficient matrix **R**. The coefficients are recorded to the Memory block of the coefficient matrix **R**. In another implementation, the Memory block of the coefficient matrix **R** can be a read-only-memory (ROM) with a predefined matrix **R**; in such a case, the Row and Column multiplexers should be excluded (together with corresponding signals from the Host Bus).

At the calculating stage, the block receives (from the Internal Bus) signals representing an element number p, a number r of the element to be updated, the step-size parameter d, and a control signal *Update/Analysis*; other signals to control (reading from)/(writing to) the Memory block of the coefficient matrix **R** can also be received from the Internal Bus. There are two operation modes of the block in the calculating stage: *Update* and *Analysis* modes. When in the *Analysis* mode, the row and column addresses of the Memory block of the coefficient matrix **R** are equal to the element number p., i.e., p^{th} diagonal element of the matrix **R** is read from the Memory block of the coefficient matrix **R**. This is provided by the Update multiplexer whose address input is connected with the signal *Update/Analysis*. When in the *Update* mode, the row address of the Memory block of the coefficient matrix **R** is r. Output of the Memory block of the coefficient matrix **R** is bit-shifted according to the step-size parameter d by means of a Bit shift block. In the *Update* mode, output data of the block are sent to the Block of Q-updates; in the *Analysis* mode, output data of the block are sent to the Block of minimisation.

At all these stages the block can also receive (from the Host Bus and/or from the Internal Bus) other signals to control the Memory block of the coefficient matrix **R**.

A variant of implementation of the block of h-updates is shown in Figure 10; this variant is for complex-valued data.

At the initialisation stage, the block receives the signal *Init* from the Host Bus; upon this signal all elements of a Memory block of the solution vector **h** are set to zero.

At the calculating stage, the block receives (from the Internal Bus) signals representing the element number p, the step-size parameter d, and signals I_1 and I_2 indicating whether the real or imaginary part of the element $h(p)$ is updated (I_2) and whether addition or subtraction of the step-size parameter d is performed (I_1). Also, a signal *Update/Reading* defines that an Update/Reading multiplexer transmits signals I_2 and p to the address input of the Memory block of the solution vector **h**. The real or imaginary part (depending of the signal I_2) of the parameter $h(p)$ is read from the Memory block of the solution vector **h**. In an Adder-subtractor, this part of the parameter $h(p)$ is updated by adding or subtracting (depending on the signal I_1) the step-size parameter d. The updated parameter $h(p)$ is re-written in the Memory block of the solution vector **h** at the same address.

At the stage of reading the results, the block receives (from the Host Bus and/or may be from the Internal Bus) signals representing an address of reading from the Memory block of the solution vector **h** (*Reading address*) and the signal *Update/Reading*. The signal *Reading address* is passed to the address input of the Memory block of the solution vector **h**, and output signals (*Elements of the solution vector*) representing the solution vector **h** are sent to the Host Bus.

At all these stages the block can also receive (from the Host Bus and/or from the Internal Bus) other signals to control the Memory block of the solution vector **h**.

A variant of implementation of the Block of Q-updates is shown in Figure 11; this variant is for complex-valued data.

At the initialisation stage, the block receives (from the Host Bus and/or from the Internal Bus) a signal *Update/Init*, which allows the initialisation data (signals representing elements of the vector β) to pass through a Data multiplexer and to be written in the Memory block of the auxiliary vector **Q**. Also, the block receives (from

the Host Bus) signals *Init address* representing an address and a signal *Update/Analysis/Init* allowing the address *Init address* to pass through an Update/Analysis/Init multiplexer to the address input of the Memory block of the auxiliary vector **Q**.

At the calculating stage, the block receives (from the Internal Bus) signals representing the element number p, the number r of the element to be updated, the step-size parameter d, and signals I_1 and I_2 indicating a type of the update of the vector **Q**. Also, the block receives (from the Block of R-memory) signals representing the value $d \cdot R(p,r)$. At this stage, the block operates in one of the following modes: *Update* mode or *Analysis* mode. In the *Analysis* mode, the Update/Analysis/Init multiplexer passes signals representing the element number p to the address input of the Memory block of the auxiliary vector **Q**; the Memory block of the auxiliary vector **Q** sends signals representing the element $Q(p)$ to the Block of minimisation. In the *Update* mode, the Update/Analysis/Init multiplexer passes signals representing the number r of the element to be updated to the address input of the Memory block of the auxiliary vector **Q**; the Memory block of the auxiliary vector **Q** sends signals representing the element $Q(r)$ to a Complex adder-subtractor. Simultaneously, signals representing the value $d \cdot R(p,r)$ are at another data input of the Complex adder-subtractor. The signals I_1 and I_2 define the type of update: if $I_1=0/1$ the Complex adder-subtractor performs summation/subtraction; the signal $I_2=0/1$ indicates that the value $d \cdot R(p,r)$ is (not multiplied)/(multiplied) by i; the latter means that real and imaginary parts of the value are replaced by each other. The updated element $Q(r)$ at the output of the complex Adder-subtractor is re-written to the Memory block of the auxiliary vector **Q** at the same address.

At all these stages the block can also receive (from the Host Bus and/or from the Internal Bus) signals to control the Memory block of the auxiliary vector **Q**.

A variant of implementation of the Block of minimisation is shown in Figure 12; this variant is for complex-valued data. The block receives (from the Block of Q-

updates) real and imaginary parts of a p^{th} element of the auxiliary vector **Q**: $a=Re[Q(p)]$ and $b=Im[Q(p)]$. The block also receives (from the Block of R-memory) a signal representing the parameter $d \cdot R(p,p)$. A Converter-a converts the signal a to two signals representing, respectively, the sign $sgn(a)$ and module $|a|$ of a. Similarly, a Converter-b converts b to $sgn(b)$ and $|b|$. First comparator compares $|a|$ and $|b|$ and forms a binary signal $I_2=0/1$ if $|a|>|b|$ or $|a|<=|b|$, respectively. A multiplexer passes the maximum of $|a|$ and $|b|$ to a Second comparator. At another input of the Second comparator are signals representing the parameter $d \cdot R(p,p)$. The second comparator compares the input signals and forms a binary signal $I_3=0/1$ if $max(|a|, |b|) > d \cdot R(p,p)$ or $max(|a|, |b|) <= d \cdot R(p,p)$, respectively. The signals representing $sgn(a)$ and $sgn(b)$ go to the data inputs of a Bit multiplexer whose address input is connected to the signal I_2; output of the Bit multiplexer forms a signal I_1.

The equality $I_3=0$ means that the current iteration is successful and the element $h(p)$ of the solution vector **h** should be updated as well as all elements of the auxiliary vector **Q** should be updated, while $I_3=1$ means that the current iteration is not successful and no update is required. The equality $I_2=0/1$ means that the real/imaginary part of the element $h(p)$ of the solution vector **h** should be updated; it also indicates that the value $d \cdot R(p,r)$ is (not multiplied)/(multiplied) by i; the latter means that real and imaginary parts of the value are replaced by each other. The equality $I_1=0/1$ means that addition/subtraction is used when updating.

The signals I_1, I_2, and I_3 are sent to the Internal Bus. From the Internal Bus, these signals are received by the Controller, Block of h-update, and Block of Q-updates. In some implementations, these signals can be modified by the Controller before the use in the Block of h-updates and Block of Q-update.

10. Multiuser receiving method and multiuser receiver

The invention is also applicable to all kinds of CDMA data transmission systems.

In the following, the invention will be described in greater detail in a cellular radio system without, however, limiting it thereto. In a cellular radio system, in which the method of multiuser receiving according to the invention can be applied, each cell comprises at least one base station communicating with mobile stations. The base station transmits calls from the mobile stations to a public telephone network or to another mobile station. A succession of data signals of a mobile user, translated into bits, is transmitted at a rate of a plurality of chips per bit, spread by a respective spreading code. All the mobile stations transmit at the same frequency to the base station, which distinguishes the transmissions of different mobile stations on the basis of respective different spreading codes. The signals of the mobile stations can also propagate along several different paths to the base station.

In the multiuser receiving method in accordance with the invention, a signal received by an antenna in a receiver is transmitted through radio frequency parts and other necessary signal processing parts, to filters matched with the spreading codes used, and in the filters the signal can be restored to the original band. The matched filters output the signals of the respective users with the respective spreading codes extracted. The signals at the matched filter outputs are grouped in a *Nx1* vector β, where N is the number of the spreading codes. Results of the matched filtering are transformed to a solution vector **h** whose values are represented by a solution of a system of linear equations of the kind $\mathbf{Rh}=\beta$ where **R** is a *NxN* cross-correlation matrix of the spreading codes and **h** is the *Nx1* solution vector. When transforming the matched-filter outputs, the method of solving systems of linear equations according to the invention is applied. The method of solving systems of linear equations can be implemented as a sequence of operations in a computer system. In another embodiment of the invention, when transforming the matched-filter output vector, the computer system according to the invention is applied. The signals representing elements of the solution vector **h** are then forwarded to a detector, where a data decision is made. The cross-correlation matrix of the spreading codes comprises

computed cross-correlations of the spreading codes used, and it indicates how much the signals of different users interfere with one another. Since the solving of the system of linear equations is performed by the method or the computer system according to the invention, which can be implemented without multiplication and division operations, the multiuser receiving is simplified.

The multiuser receiver in accordance with the invention can be implemented as represented in Figure 13. Figure 13 is a block diagram illustrating the basic structure of the multiuser receiver according to the invention. The multiuser receiver comprises an antenna with which a received signal is forwarded to radio frequency (RF) parts, where the radio frequency signal is converted to an intermediate frequency. From the radio frequency parts, the signal is supplied to an analog-to-digital converter (A/D), where the received analog signal is converted to digital form. The converted digital signal is supplied to detector means (DET), and the output signal is then forwarded to the other parts of the receiver. The receiver also comprises control means (CNTL), which control the operation of the above-mentioned blocks.

Figure 14 illustrates in greater detail implementation of the detector means according to one preferred embodiment of the invention. The detector block comprises filters matched with the spreading codes, in the matched filters, the spread spectrum signal is restored to the original narrow band. The number of matched filters is equal to the number of signals that are to be used in the detection, such as the number of users N. Signals obtained from the matched filters are supplied to a computer system and transformed into a solution vector $\mathbf{h}$ whose elements are represented by a solution of a system of linear equations of the kind $\mathbf{Rh}=\beta$ where $\mathbf{R}$ is a NxN cross-correlation matrix of the spreading codes and $\mathbf{h}$ is the $Nx1$ solution vector. When transforming the matched-filter output vector, the method of solving systems of linear equations according to the invention is applied; the method of solving systems of linear equations can be implemented as a sequence of operations in a computer system. In another embodiment of the invention, when transforming the

signals obtained from the matched-filter, the computer system according the invention can be applied. The signals representing elements of the vector **h** are then forwarded to a means making a data symbol decision. The receiver naturally also comprises other components, such as filters or means for calculating the cross-correlation matrix, but for the sake of clarity and since they are not essential to the present invention, they have not been shown.

11. Adaptive filtering

As an example, we consider adaptive filtering in application to echo cancellation in a transmission system. However, other applications of adaptive filtering according to the invention are also possible. Prior to description of the preferred embodiment, description will be made as to general arrangement and operation of the data transmission system with an echo canceller so as to help better understanding of he present invention.

Referring to Figure 15, an incoming signal propagates as a first signal $x(t)$ through a first transmission line to an unknown system which produces a second signal $y(t)$ transmitted to a second transmission line. Ideally, the second signal $y(t)$ contains only a signal of interest $s(t)$: $y(t)=s(t)$, and does not contain any echo: $e(t)=0$. In practice, however, a part of the first signal $x(t)$ leaks to the second transmission line through the unknown system. Then, the signal of interest $s(t)$ is mixed with the echo to form the second signal: $y(t)=s(t)+e(t)$. The echo $e(t)$ should be cancelled from the second signal so as to transmit the second signal as the signal of interest $s(t)$ alone. The unknown system can be a hybrid transformer; a transmission system with a hybrid transformer and echo cancellation is discussed in greater detail in the Patent US 5062102, "Echo canceller with means for determining filter coefficients from autocorrelation and cross-correlation coefficients" by Tetsu Taguchi. The unknown system can also be an acoustic system as, for example, is discussed in greater detail in Patent WO 00/38319, "Stable adaptive filter and method" by Ding Heping.

In order to cancel the echo from the second signal, the echo canceller comprises a filter such as a transversal filter coupled to the first transmission line and responsive to the first signal $x(t)$ for producing an echo replica or an estimated echo signal $e_1(t)$ determined in accordance with the filter coefficients $h(i)$ $(i=1,2,...,N)$. A subtractor is coupled to the filter and connected in the second transmission line for subtracting the estimated echo signal $e_1(t)$ from the second signal $y(t)=s(t)+e(t)$ on the second transmission line so that the echo is reduced or cancelled: $y_1(t)=y(t)-e_1(t)=s(t)+[e(t)-e_1(t)]$. The echo canceller also comprises a coefficient generating circuit for generating the filter coefficients $h(i)$ so as to determine the estimated echo signal $e_1(t)$. If the coefficient generating circuit can generate the optimum filter coefficients so as to provide $e(t)=e_1(t)$, the echo is completely cancelled. When $e(t)\neq e_1(t)$, $[e(t)-e_1(t)]$ still remains as a residual echo. That is, the echo cancellation is defined by performance of the coefficient generating circuit.

In the shown embodiment in Figure 16, the coefficient generating circuit comprises a first, a second, and a third circuit. The first circuit is coupled to the first transmission line and calculates autocorrelation coefficients of the first signal $x(t)$. The second circuit is coupled to the first and the second transmission lines and calculates cross-correlation coefficients between the first signal $x(t)$ and the second signal $y(t)$. Accordingly, the first and the second circuits are an autocorrelation coefficient calculator and a cross-correlation coefficient calculator, respectively.

The third circuit is coupled to the autocorrelation coefficient calculator and the cross-correlation coefficient calculator and decides the filter coefficients $h(i)$ $(i=1,2,...,N)$. The third circuit is called a filter coefficient decision circuit. It is generally known in the prior art that the optimal filter coefficients are that obtained by solving a system of linear equations of the kind $\mathbf{Rh}=\beta$ where $\mathbf{R}$ is a NxN coefficient matrix comprising the autocorrelation coefficients $R(m,n)$ of the first signal x(t),

$$R(m,n)=\sum_{t=1}^{T-|n-m|}x(t)x(t+|m-n|),\quad m,n=1,2,...,N,$$

where $t=1,...,T$ are discrete time moments, a $Nx1$ vector β comprises the cross-

correlation coefficients,

$$\beta(n) = \sum_{t=1}^{T-n-1} x(t)y(t+n-1)$$

and **h** is a *Nx1* solution vector comprising the filter coefficients *h(i) (i=1,2,...,N)*; this is discussed in greater detail in the Patent US 5062102, "Echo canceller with means for determining filter coefficients from autocorrelation and cross-correlation coefficients" by Tetsu Taguchi.

The autocorrelation coefficients and the cross-correlation coefficients are provided to the third circuit from the autocorrelation coefficient calculator and the cross-correlation coefficient calculator, respectively. The third circuit in the embodiment is a computer system for solving systems of linear equations. When the computer system is implemented by using the described above method for solving systems of linear equations in accordance with the invention, the convergence speed increases and truncation errors are eliminated. As a result, the echo canceller exploiting such a computer system allows better cancellation of the echo. In another embodiment, the computer system is implemented as the computer system in accordance with the invention; this allows the filter coefficients to be calculated without multiplication or division operations, leading to simpler implementation. The latter allows the sampling frequency of the adaptive filter to be increased and to be applied in data transmission systems with higher data rates. This also allows the number of filter coefficients *N* to be increased, providing a better echo cancellation, i.e., a smaller residual echo.

Although the invention has been described above with reference to examples illustrated in the attached drawings, it is to be understood that the invention is not limited thereto but can be modified in many ways within the scope of the inventive idea claimed in the following claims.

Claims

1. A method for solving a system of linear equations, comprising the steps of:
    a. representing elements of a solution vector as fixed-point binary words each consisting of at least one bit;
    b. initialising the solution vector and an auxiliary vector;
    c. performing, for each bit representing the binary words, bit-wise iterations comprising the steps of:
        i. performing passes through all elements of the solution vector;
        ii. updating elements of the solution vector in the passes;
        iii. updating elements of the auxiliary vector in the passes;
        iv. repeating the passes until a finishing condition is fulfilled;
    d. stopping solving the system of linear equations when a stopping condition is fulfilled.
2. The method as defined in claim 1 wherein elements of the solution vector are initialised as zeros.
3. The method as defined in claim 1 wherein the auxiliary vector is initialised by the right-side vector of the system of linear equations.
4. The method as defined in claim 1 wherein the bit-wise iterations start from the most significant bit and proceed with the next less significant bit if the finishing condition is fulfilled.
5. The method as defined in claim 1 wherein in each pass, in turn, for each element of the solution vector a condition successful/unsuccessful is checked.
6. The method as defined in claim 1 wherein the finishing condition is fulfilled if in a pass no element of the solution vector is updated.
7. The method as defined in claim 1 wherein the stopping condition is fulfilled if a predefined number of passes through all elements of the solution vector is exceeded.
8. The method as defined in claim 1 wherein the stopping condition is fulfilled if a predefined number of bit-wise iterations, defining the number of valuable

bits in elements of the solution vector as well as accuracy of the solution, is exceeded.

9. The method as defined in claim 1 wherein the stopping condition is fulfilled if a computer time predefined for performing this method is finished.

10. The method as defined in claim 1 wherein when passing through elements of the solution vector the order of analysing elements of the solution vector in the pass is arbitrary.

11. The method as defined in claim 1 wherein when passing through elements of the solution vector the pass starts from an element whose position corresponds to the position of an element of the auxiliary vector with maximum amplitude and in the order of reducing the amplitude.

12. The method as defined in claim 11 wherein ordering elements of the auxiliary vector is performed to define the order of elements in the pass.

13. The method as defined in claim 5 wherein updating elements of the solution vector and the auxiliary vector is performed only if the condition successful/unsuccessful is successful.

14. The method as defined in claim 13 wherein the only element of the solution vector is updated, for which the condition successful/unsuccessful is checked.

15. The method as defined in claim 14 wherein a finite number of possible updates of the element of the solution vector are analysed for finding a preferable update and the element of the solution vector when updated is set to be equal to the preferable update.

16. The method as defined in claim 15 wherein finding the preferable update comprises the steps of:

    a. calculating, for each possible update, an auxiliary value;
    b. finding a minimum among the auxiliary values;
    c. calculating a threshold;
    d. comparing the minimum with the threshold;
    e. choosing the preferable update as that corresponding to the minimum.

17. The method as defined in claim 16 wherein the condition successful/unsuccessful is successful if the minimum among the auxiliary values is less than the threshold, and the condition successful/unsuccessful is unsuccessful if the minimum among the auxiliary values is higher than or equal to the threshold.

18. The method as defined in claim 16 wherein calculating the auxiliary values is based on the corresponding element of the auxiliary vector.

19. The method as defined in claim 16 wherein calculating the threshold is performed by using a diagonal element of the coefficient matrix, the diagonal element corresponding to the element of the solution vector, and a step-size parameter.

20. The method as defined in claim 19 wherein the step-size parameter is decreased after each bit-wise iteration.

21. The method as defined in claim 20 wherein decreasing the step-size is by a factor of two.

22. The method as defined in claim 13 wherein elements of the auxiliary vector are updated by using elements of the coefficient matrix and the step-size parameter.

23. The method as defined in claim 22 wherein an element of the auxiliary vector is updated by using elements of a row of the coefficient matrix, the row corresponding to the updated element of the auxiliary vector, and the step-size parameter.

24. A computer system for solving a system of linear equations, comprising:

    a. a host processor producing itself or receiving from other devices parameter signals representing elements of a coefficient matrix and right-side vector of the system of linear equations and transmitting to other devices parameter signals representing elements of a solution vector;

    b. a host bus coupled to the host processor;

    c. an internal bus;

    d. a first means for storing and updating elements of the solution vector, the first means coupled to the host bus and the internal bus;

    e. a second means for storing elements of the coefficient matrix, the second means coupled to the host and internal buses;

    f. a third means for determining successful iterations and preferable updates, the third means coupled to the internal bus and the second means;

g. a fourth means for storing and updating elements of an auxiliary vector, the fourth means coupled to the host bus, the internal bus, and the second means.

25. The computer system of claim 24 comprising a controller coupled to the host bus and the internal bus, receiving control signals from the host processor through the host bus and producing control signals for the internal bus.

26. The computer system of claim 24, wherein the first means contains a memory means for storing elements of the solution vector and a means for adding or subtracting, and the first means updates elements of the solution vector by adding or subtracting a step-size parameter.

27. The computer system of claim 24, wherein the second means contains a memory means for storing elements of the coefficient matrix and a means for bit-shifting, and the second means performs bit-shifts of elements of the coefficient matrix.

28. The computer system of claims 24, wherein the fourth means contains a memory means for storing elements of the auxiliary vector and a means for adding or subtracting, and the fourth means updates elements of the auxiliary vector by adding or subtracting bit-shifted elements of the coefficient matrix.

29. The computer system of claim 24, wherein the fourth means receives initialisation data from the host bus.

30. The computer system of claim 24, wherein the fourth means receives initialisation data from the host bus, the initialisation data being elements of the right-side vector of the system of linear equations.

31. The computer system of claim 24, wherein the second means receives elements of the coefficient matrix from the host bus.

32. The computer system of claim 24, wherein the third means determines successful iterations by comparing elements of the auxiliary vector and bit-shifted elements of the coefficient matrix.

33. A multiuser receiving method in a data transmission system in which code division multiple access, involving multiuser interference among respective signals, each signal representing a succession of data signals translated into bits and transmitted at a rate of a plurality of chips per bit, spread by a respective spreading code, is applied for detecting a particular data signal, from among a plurality of data signals, said method comprising:

a. filtering matched with the spreading codes and applied to the received signal to obtain respective output signals;

b. transforming the matched-filter output signals by solving a system of linear equations of the kind $\mathbf{Rh}=\beta$ where $\mathbf{R}$ is a NxN cross-correlation matrix of the spreading codes, β is a $Nx1$ vector grouping the matched-filter output signals, $\mathbf{h}$ is the $Nx1$ solution vector representing the transformed signals, and N is a number of used spreading codes, wherein solving the system of linear equations comprises the steps of:

i. representing elements of a solution vector as fixed-point binary words each comprising at least one bit;

ii. initialising the solution vector and an auxiliary vector;

iii. performing, for each bit representing the binary words, bit-wise iterations comprising the steps of: performing passes through all elements of the solution vector; updating elements of the solution vector in the passes; updating elements of the auxiliary vector in the passes; repeating the passes until a finishing condition is fulfilled;

iv. stopping solving the system of linear equations when a stopping condition is fulfilled.

c. subjecting the transformed signals to obtain estimates of the data signals.

34. A multiuser receiver in a data transmission system in which code division multiple access, involving multiuser interference among respective signals, each signal representing a succession of data signals translated into bits and transmitted at a rate of a plurality of chips per bit, spread by a respective spreading code, is applied for detecting a particular data signal, from among a plurality of data signals, said multiuser receiver comprising:

a. filters matched with spreading codes contained in the received signals;

b. a computer system for solving systems of linear equations of the kind $\mathbf{Rh}=\beta$ where $\mathbf{R}$ is a NxN cross-correlation matrix of the spreading codes, β is a $Nx1$ vector grouping the matched-filter output signals, $\mathbf{h}$

is the *Nx1* solution vector representing the output signals of the computer system, and *N* is a number of used spreading codes, wherein the computer system performs a sequence of operations comprising the steps of:

i. representing elements of a solution vector as fixed-point binary words each consisting at least one bit;

ii. initialising the solution vector and an auxiliary vector;

iii. performing, for each bit representing the binary words, bit-wise iterations comprising the steps of: performing passes through all elements of the solution vector; updating elements of the solution vector in the passes; updating elements of the auxiliary vector in the passes; repeating the passes until a finishing condition is fulfilled;

iv. stopping solving the system of linear equations when a stopping condition is fulfilled.

c. a means for estimating the data signals from the output signals of the computer system.

35. A multiuser receiver according to claim 34, wherein the computer system for solving the system of linear equations comprises:

a. a host processor receiving parameter signals representing elements of the cross-correlation matrix of the spreading codes and right-side vector of the system of linear equations and transmitting parameter signals representing elements of the solution vector;

b. a host bus coupled to the host processor;

c. an internal bus;

d. a first means for storing and updating elements of the solution vector, the first means coupled to the host bus and the internal bus;

e. a second means for storing elements of the cross-correlation matrix, the second means coupled to the host bus and the internal bus;

f. a third means for determining successful iterations and preferable updates, the third means coupled to the internal bus and the second means;

g. a fourth means for updating and storing elements of an auxiliary vector, the fourth means coupled to the host bus, the internal bus, and the second means.

36. An adaptive filter for receiving a first signal from a first transmission line and a second signal from a second transmission line, said first signal partially leaking from said first transmission line to said second transmission line as an echo, said adaptive filter comprising a filter means coupled to said first transmission line and responsive to said first signal for producing an estimated echo signal determined in accordance with filter coefficients, a coefficient generating means for generating said filter coefficients, and subtracting means coupled to said filter means and connected in said second transmission line for subtracting said estimated echo signal from said second signal on said second transmission line so as to cancel said echo signal, said coefficient generating means comprises:

a. a first means coupled to said first transmission line and responsive to said first signal for producing a series of autocorrelation coefficients of said first signal;

b. a second means coupled to said first and said second transmission lines and responsive to said first and said second signal for producing a series of cross-correlation coefficients between said first signal and said second signal; and

c. a means coupled to said first and said second means for generating said filter coefficients from said autocorrelation and cross-correlation coefficients to deliver said filter coefficients to said filter means, said third means is a computer system for solving a system of linear equations whose coefficient matrix comprises said autocorrelation coefficients and whose right-side vector comprises said cross-correlation coefficients, wherein said computer system for solving the system of linear equations performs a sequence of operations comprising the steps of:

i. representing elements of a solution vector as fixed-point binary words each consisting of at least one bit;

ii. initialising the solution vector and an auxiliary vector;

iii. performing, for each bit representing the binary words, bit-wise iterations comprising the steps of: performing passes through all elements of the solution vector; updating elements of the solution vector in the passes; updating elements of the auxiliary vector in the passes; repeating the passes until a finishing condition is fulfilled;

iv. stopping solving the system of linear equations when a stopping condition is fulfilled.

37. The adaptive filter of claim 36, wherein the computer system for solving the system of linear equations comprises:

a. a host processor receiving parameter signals representing elements of the coefficient matrix and the right-side vector and transmitting parameter signals representing elements of the solution vector;

b. a host bus coupled to the host processor;

c. an internal bus;

d. a first means for storing and updating elements of the solution vector, the first means coupled to the host bus and the internal bus;

e. a second means for storing elements of the coefficient matrix, the second means coupled to the host bus and the internal bus;

f. a third means for determining successful iterations and preferable updates, the third means coupled to the internal bus and the second means;

g. a fourth means for storing and updating elements of an auxiliary vector, the fourth means coupled to the host bus, internal bus, and the second means.

38. The adaptive filter of claim 36, wherein said filter means is a transversal filter.

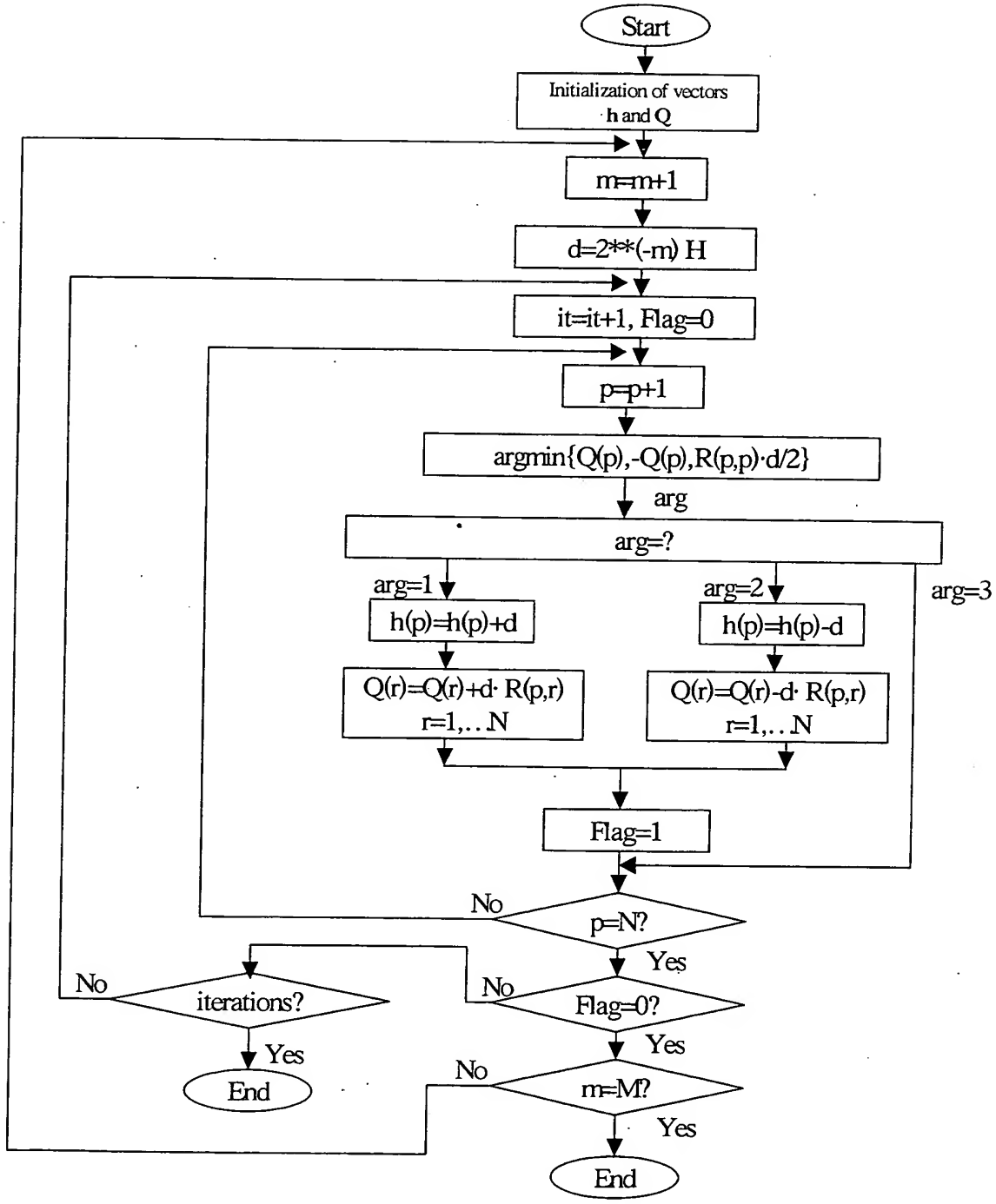
Start
Initialization of vectors h and Q
m=m+1
d=2**(-m) H
it=it+1, Flag=0
p=p+1
argmin{Q(p),-Q(p),R(p,p)·d/2}
arg
arg=?
arg=1
h(p)=h(p)+d
Q(r)=Q(r)+d· R(p,r) r=1,..N
arg=2
h(p)=h(p)-d
Q(r)=Q(r)-d· R(p,r) r=1,..N
arg=3
Flag=1
No
p=N?
Yes
No
iterations?
Yes
End
No
Flag=0?
Yes
No
m=M?
Yes
End

Figure 1

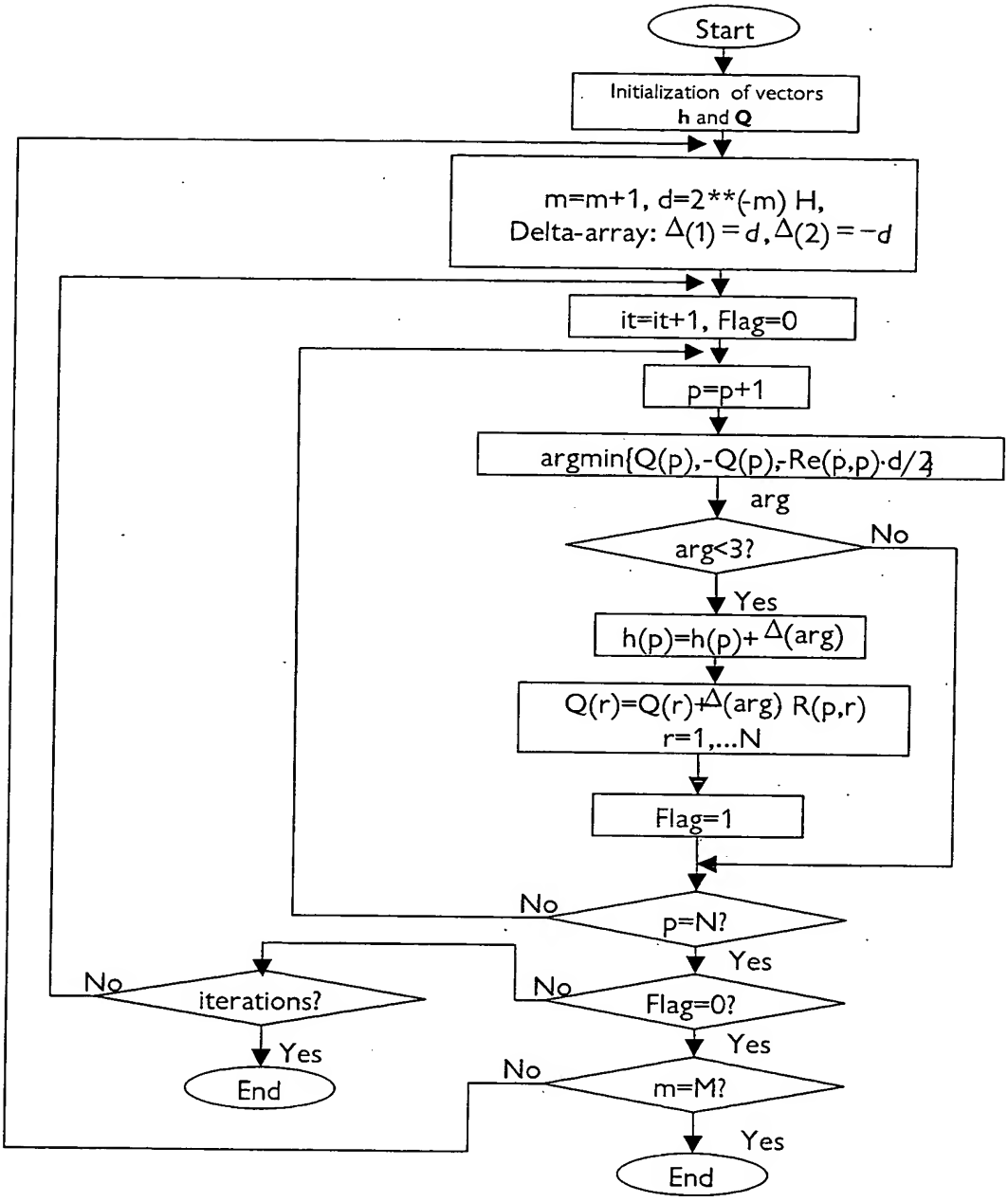
Start
Initialization of vectors
h and Q
m=m+1, d=2**(-m) H,
Delta-array: Δ(1) = d, Δ(2) = −d
it=it+1, Flag=0
p=p+1
argmin{Q(p),-Q(p),-Re(p,p)·d/2}
arg
arg<3?
No
Yes
h(p)=h(p)+ Δ(arg)
Q(r)=Q(r)+Δ(arg) R(p,r)
r=1,...N
Flag=1
No
p=N?
Yes
No
iterations?
No
Flag=0?
Yes
Yes
End
No
m=M?
Yes
End

Figure 2

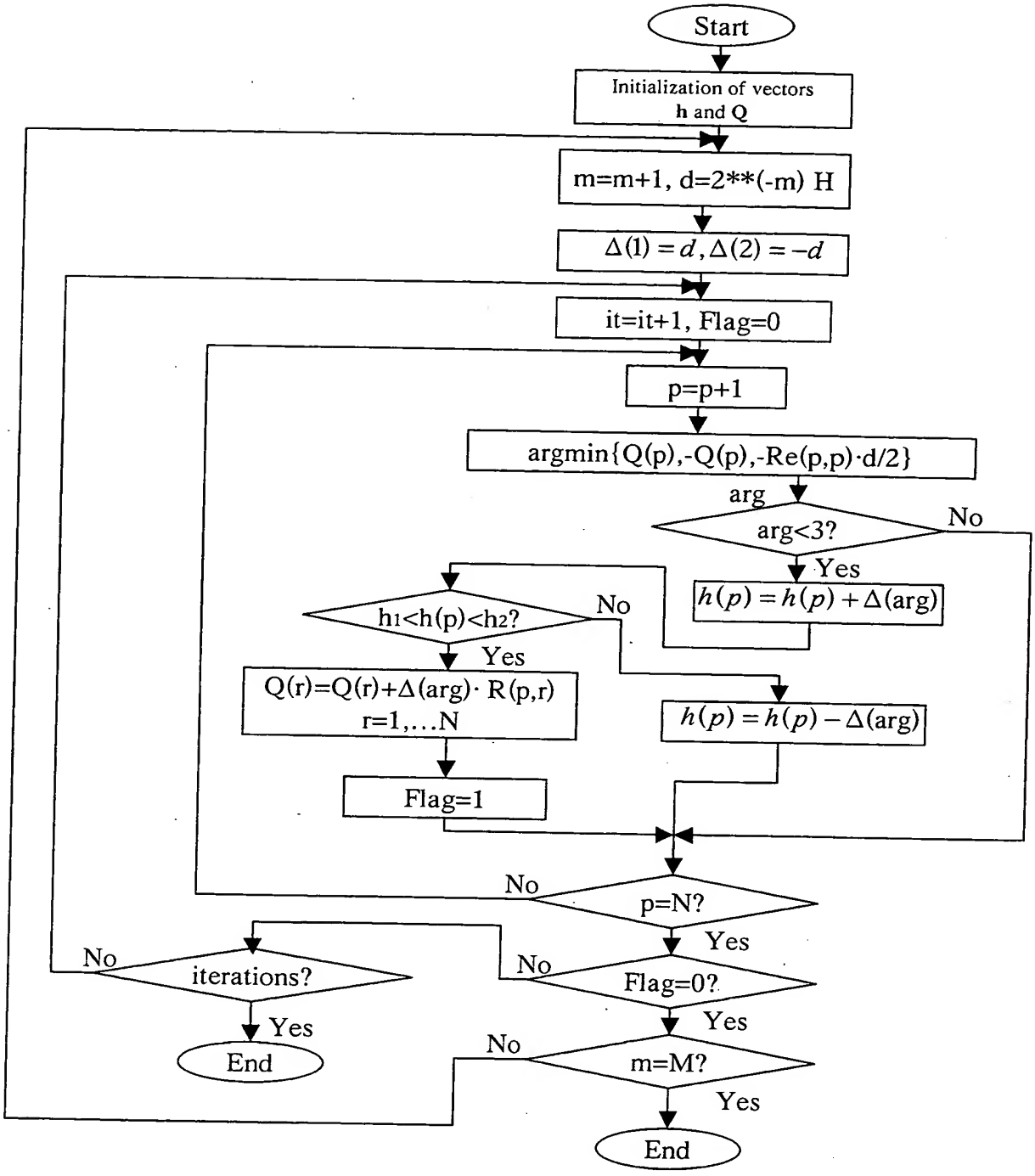
Start
Initialization of vectors h and Q
m=m+1, d=2**(-m) H
Δ(1) = d, Δ(2) = −d
it=it+1, Flag=0
p=p+1
argmin{Q(p),-Q(p),-Re(p,p)·d/2}
arg
arg<3?
No
Yes
h(p) = h(p) + Δ(arg)
h1<h(p)<h2?
No
Yes
Q(r)=Q(r)+Δ(arg)· R(p,r)
r=1,...N
h(p) = h(p) − Δ(arg)
Flag=1
No
p=N?
Yes
No
iterations?
Yes
End
No
Flag=0?
Yes
No
m=M?
Yes
End

Figure 3

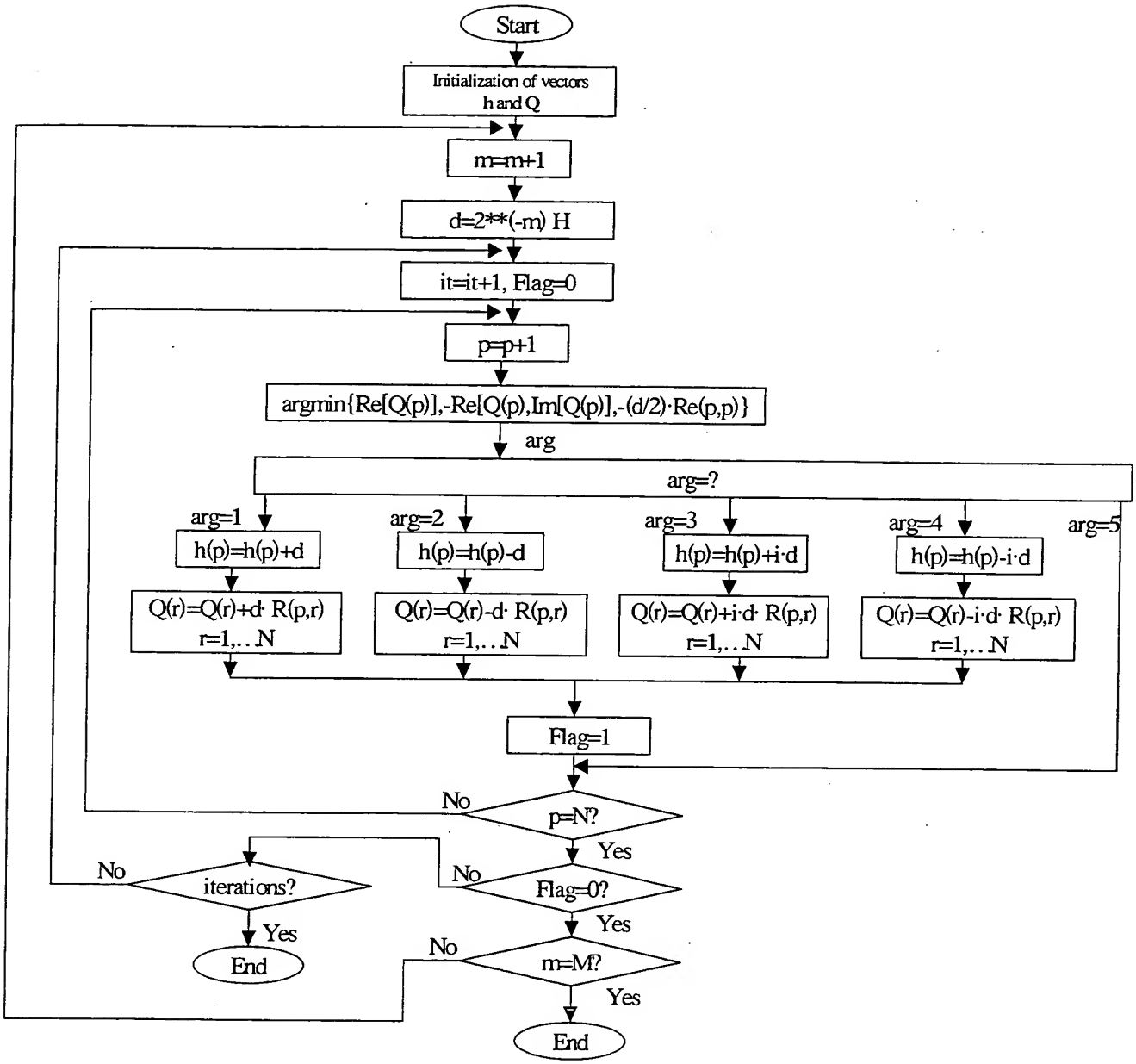
Start
Initialization of vectors h and Q
m=m+1
d=2**(-m) H
it=it+1, Flag=0
p=p+1
argmin{Re[Q(p)],-Re[Q(p),Im[Q(p)],-(d/2)·Re(p,p)}
arg
arg=?
arg=1
h(p)=h(p)+d
Q(r)=Q(r)+d· R(p,r)
r=1,..N
arg=2
h(p)=h(p)-d
Q(r)=Q(r)-d· R(p,r)
r=1,..N
arg=3
h(p)=h(p)+i·d
Q(r)=Q(r)+i·d· R(p,r)
r=1,..N
arg=4
h(p)=h(p)-i·d
Q(r)=Q(r)-i·d· R(p,r)
r=1,..N
arg=5
Flag=1
No
p=N?
Yes
No
iterations?
Yes
End
No
Flag=0?
Yes
No
m=M?
Yes
End

Figure 4

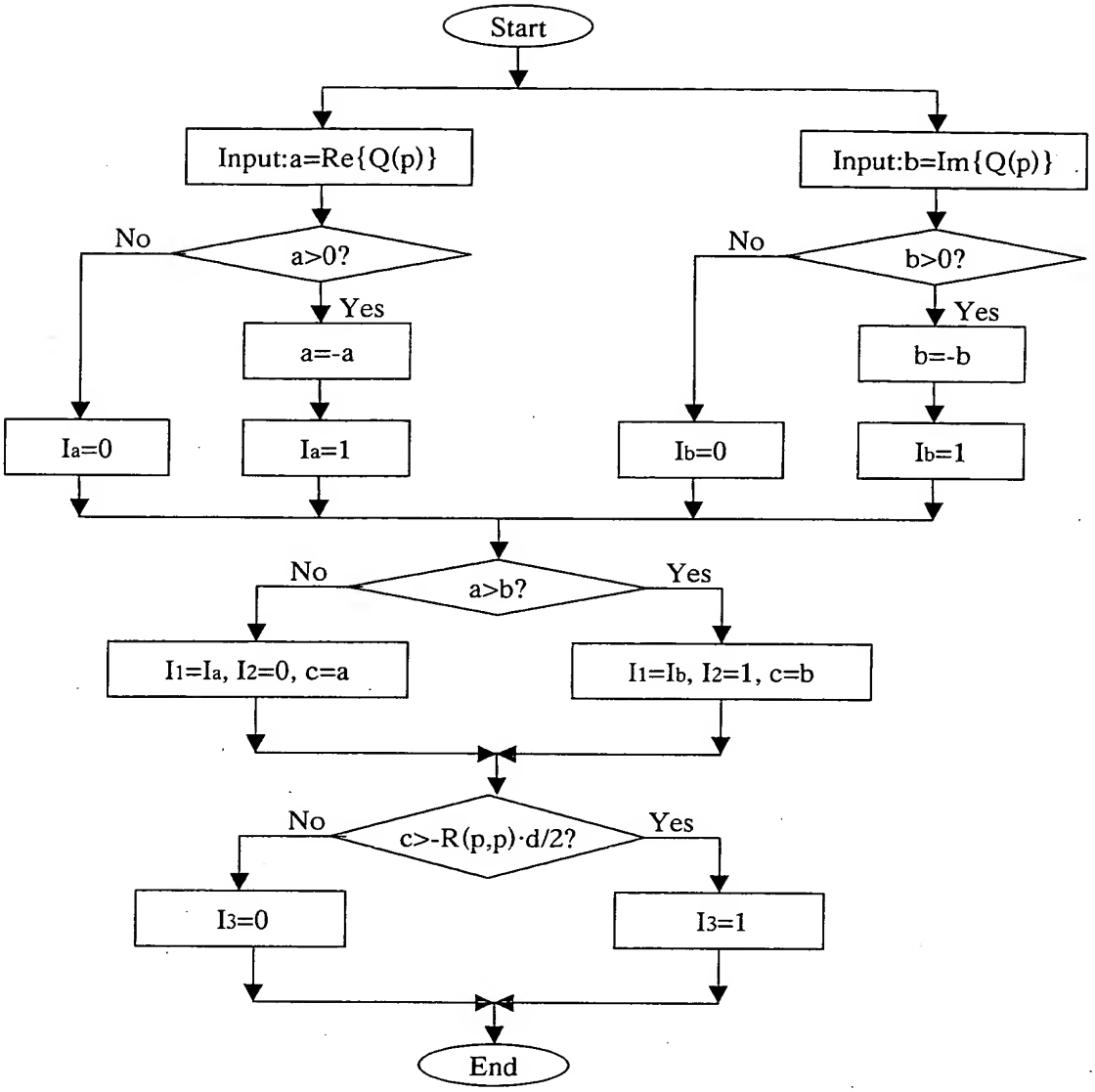
Start
Input:a=Re{Q(p)}
Input:b=Im{Q(p)}
No
a>0?
Yes
a=-a
Ia=0
Ia=1
No
b>0?
Yes
b=-b
Ib=0
Ib=1
No
a>b?
Yes
I1=Ia, I2=0, c=a
I1=Ib, I2=1, c=b
No
c>-R(p,p)·d/2?
Yes
I3=0
I3=1
End

Figure 5

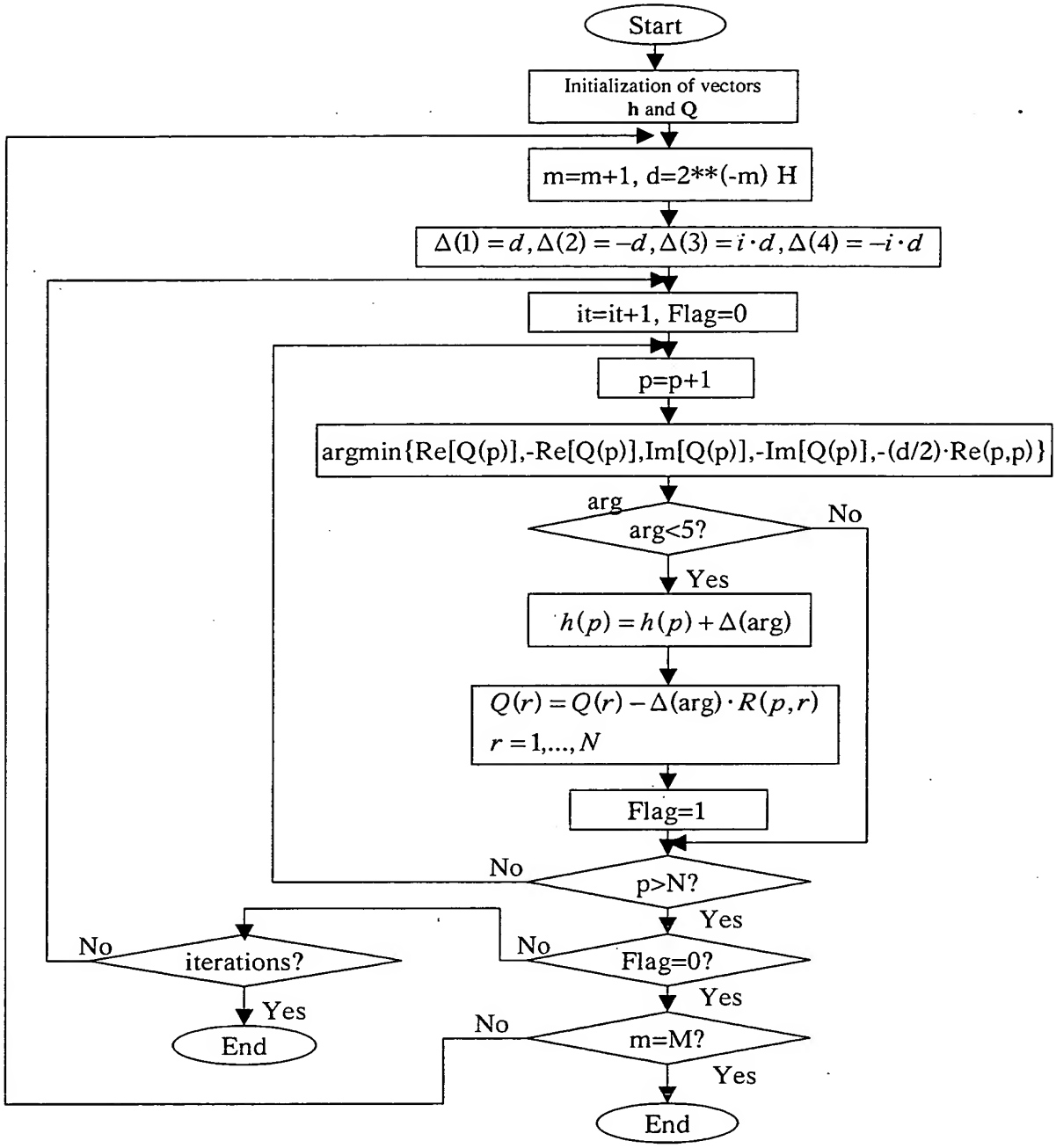
Start
Initialization of vectors h and Q
m=m+1, d=2**(-m) H
$\Delta(1)=d, \Delta(2)=-d, \Delta(3)=i\cdot d, \Delta(4)=-i\cdot d$
it=it+1, Flag=0
p=p+1
argmin{Re[Q(p)],-Re[Q(p)],Im[Q(p)],-Im[Q(p)],-(d/2)·Re(p,p)}
arg
arg<5?
No
Yes
$h(p)=h(p)+\Delta(\arg)$
$Q(r)=Q(r)-\Delta(\arg)\cdot R(p,r)$
$r=1,\ldots,N$
Flag=1
No
p>N?
Yes
No
iterations?
No
Flag=0?
Yes
Yes
End
No
m=M?
Yes
End

Figure 6

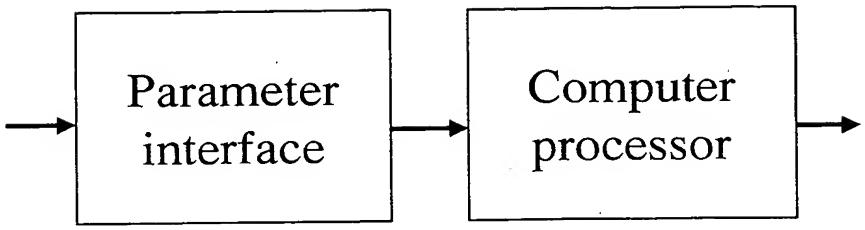

Figure 7

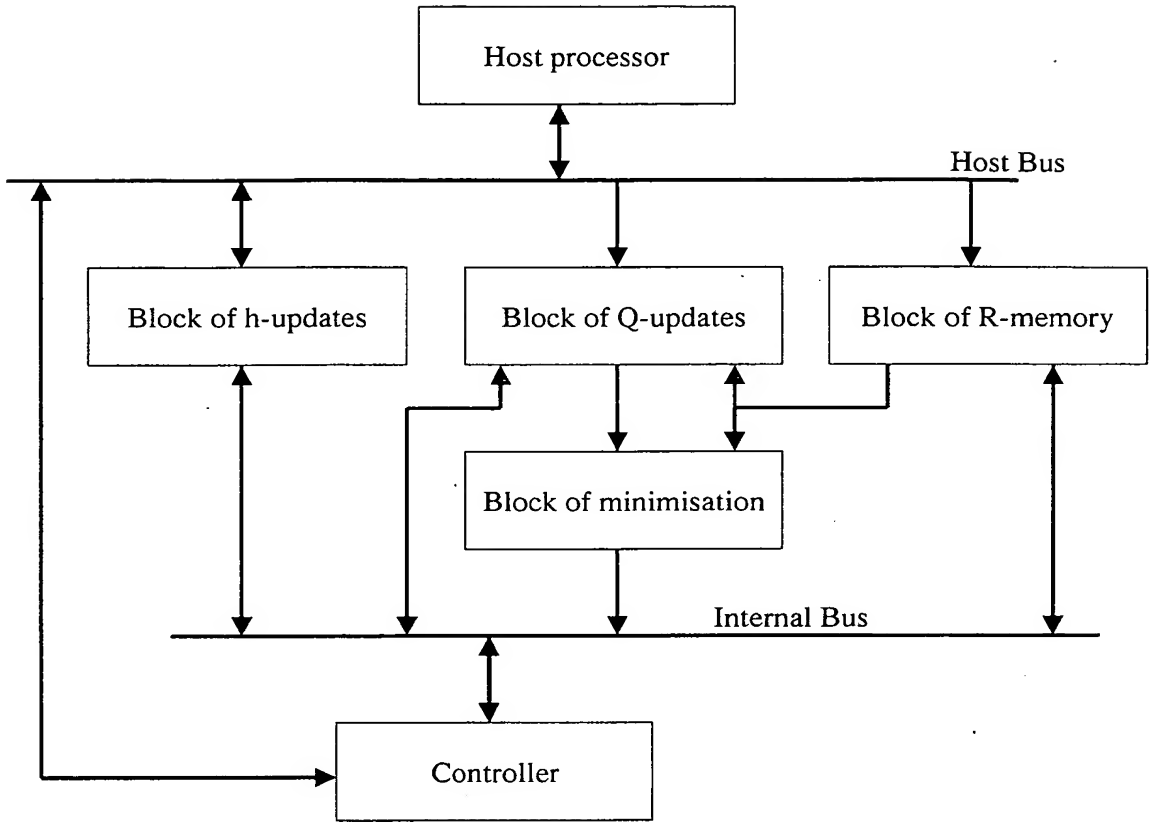
Host processor
Host Bus
Block of h-updates
Block of Q-updates
Block of R-memory
Block of minimisation
Internal Bus
Controller

Figure 8

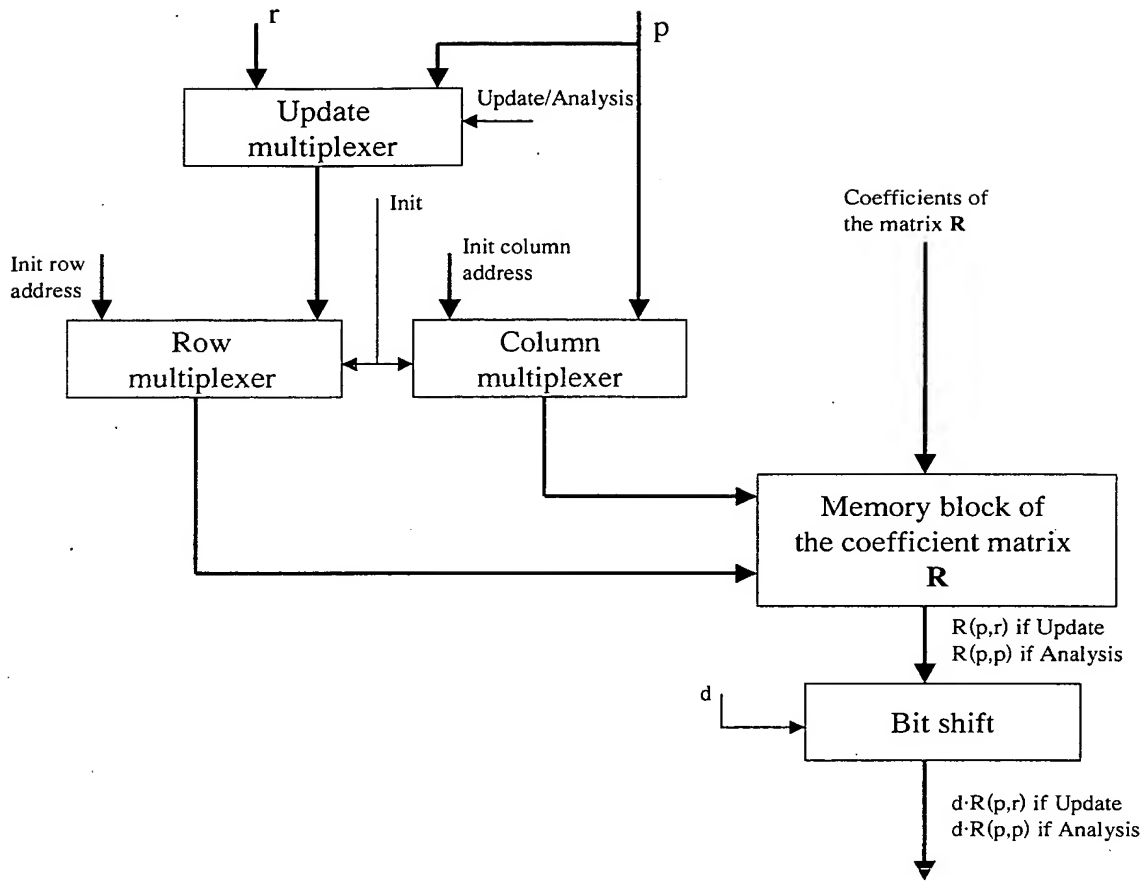
r
p
Update multiplexer
Update/Analysis
Init
Init row address
Init column address
Coefficients of the matrix R
Row multiplexer
Column multiplexer
Memory block of the coefficient matrix R
R(p,r) if Update
R(p,p) if Analysis
d
Bit shift
d·R(p,r) if Update
d·R(p,p) if Analysis

Figure 9

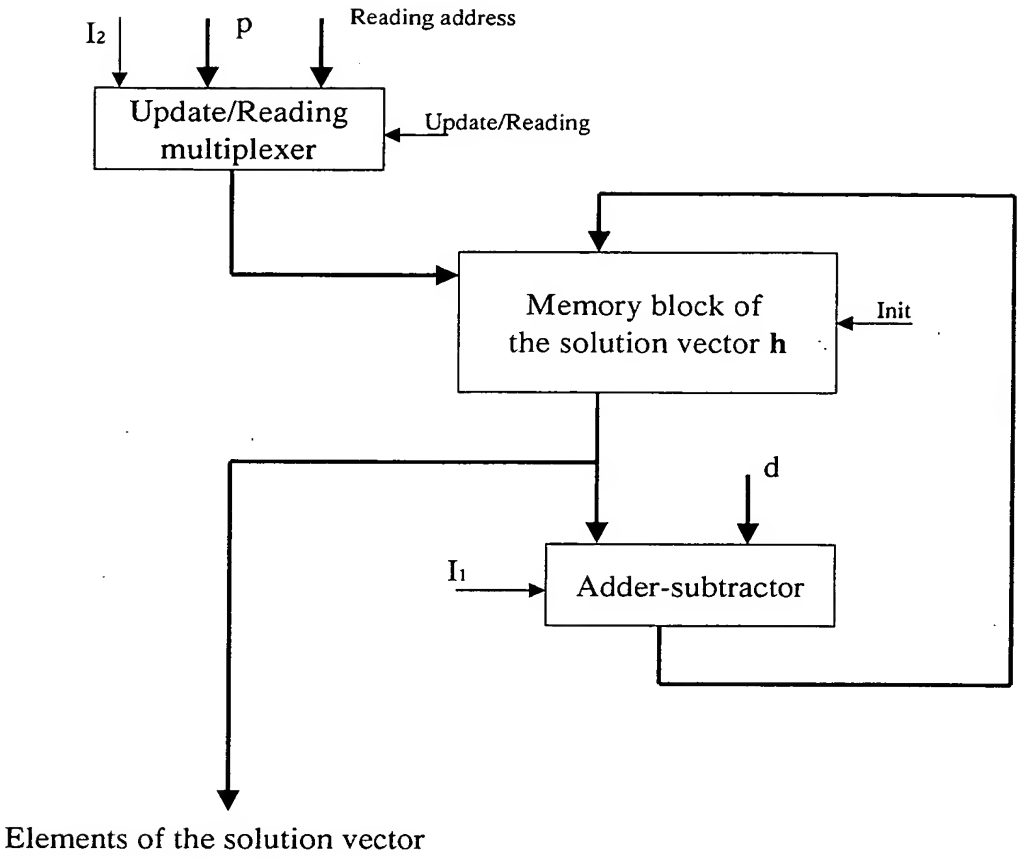
I2
p
Reading address
Update/Reading multiplexer
Update/Reading
Memory block of the solution vector h
Init
d
I1
Adder-subtractor
Elements of the solution vector

Figure 10

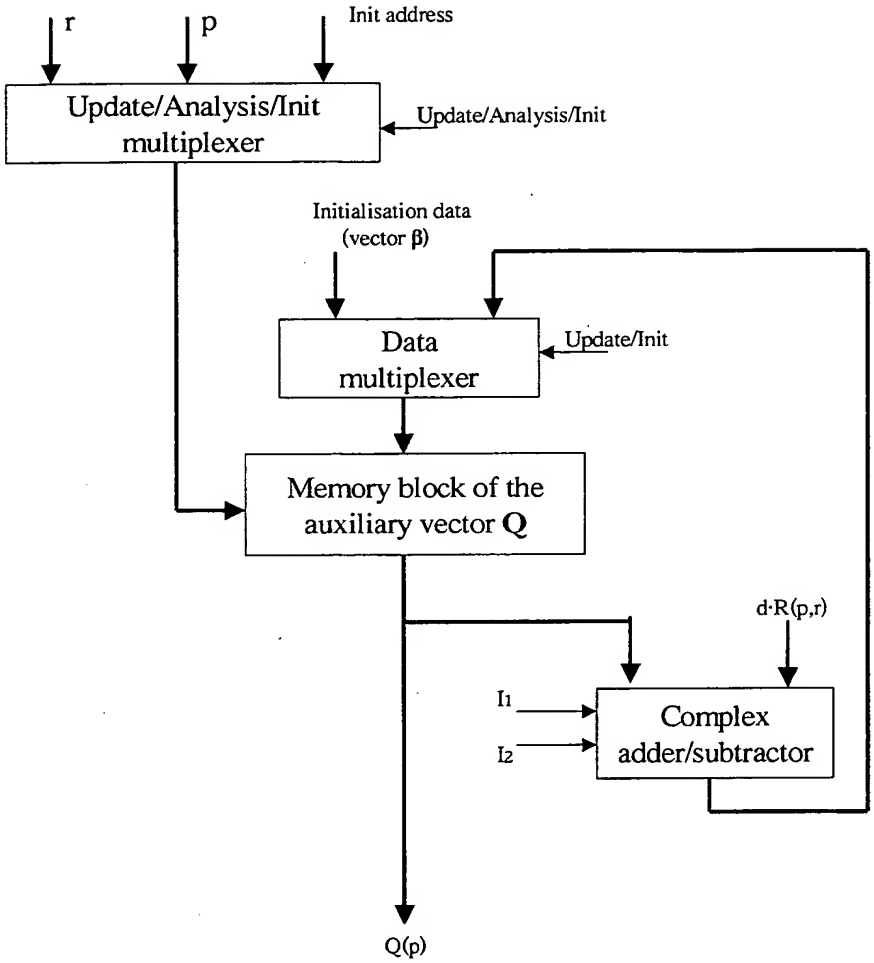
r
p
Init address
Update/Analysis/Init multiplexer
Update/Analysis/Init
Initialisation data (vector β)
Data multiplexer
Update/Init
Memory block of the auxiliary vector Q
d·R(p,r)
I1
I2
Complex adder/subtractor
Q(p)

Figure 11

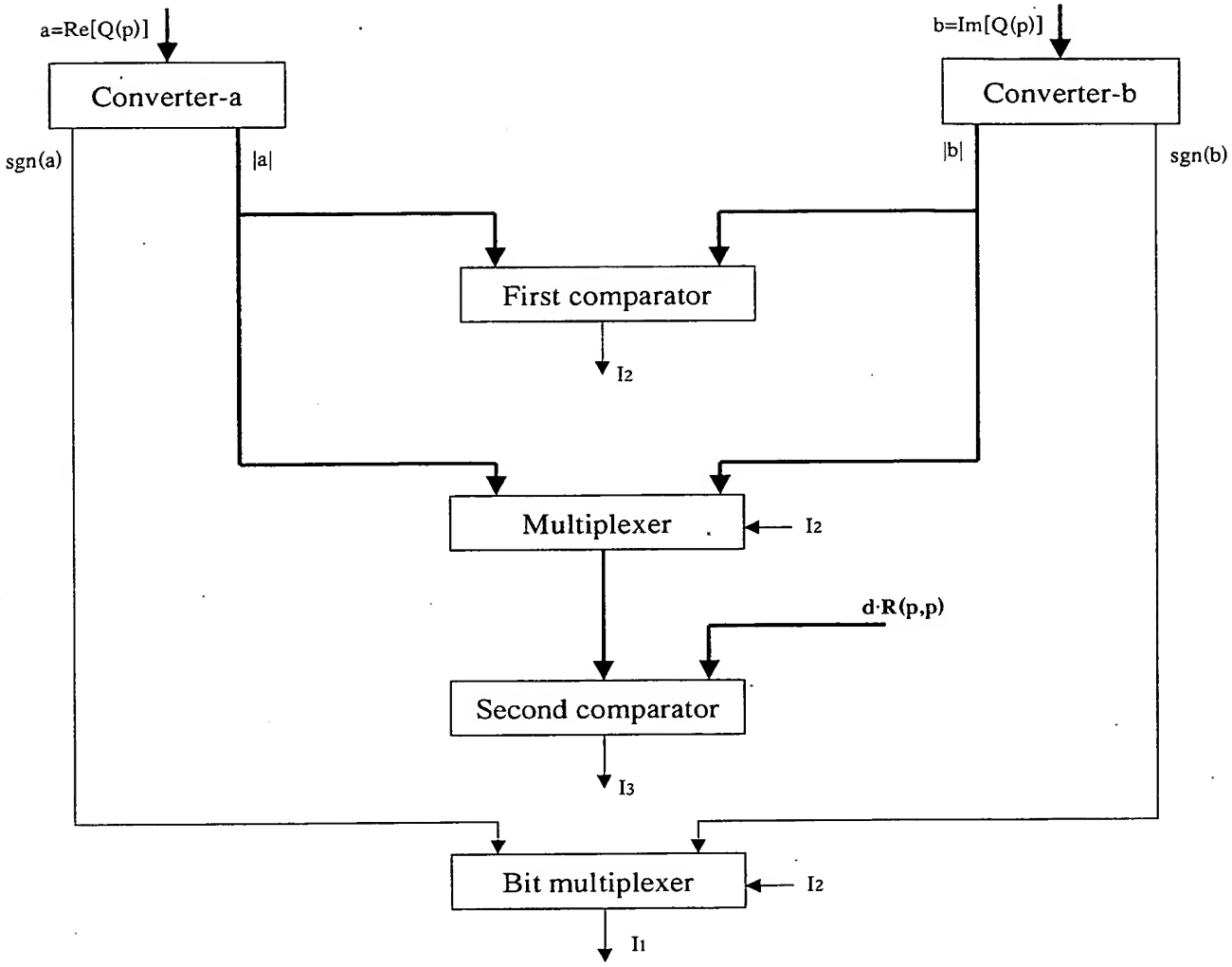
a=Re[Q(p)]
b=Im[Q(p)]
Converter-a
Converter-b
sgn(a)
|a|
|b|
sgn(b)
First comparator
I2
Multiplexer
I2
d·R(p,p)
Second comparator
I3
Bit multiplexer
I2
I1

Figure 12

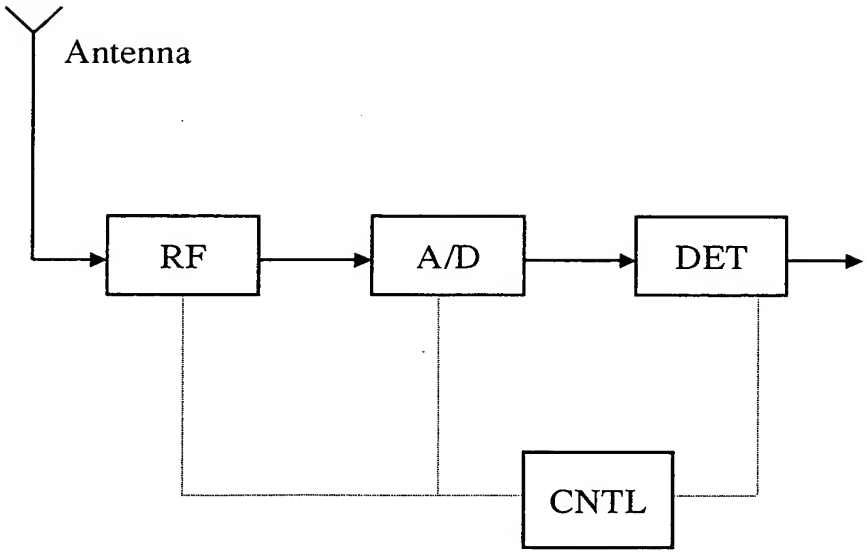

Figure 13

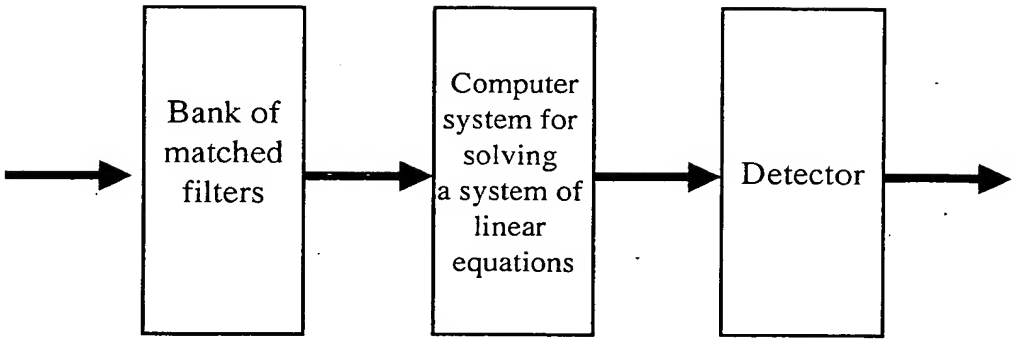
Bank of matched filters
Computer system for solving a system of linear equations
Detector

Figure 14

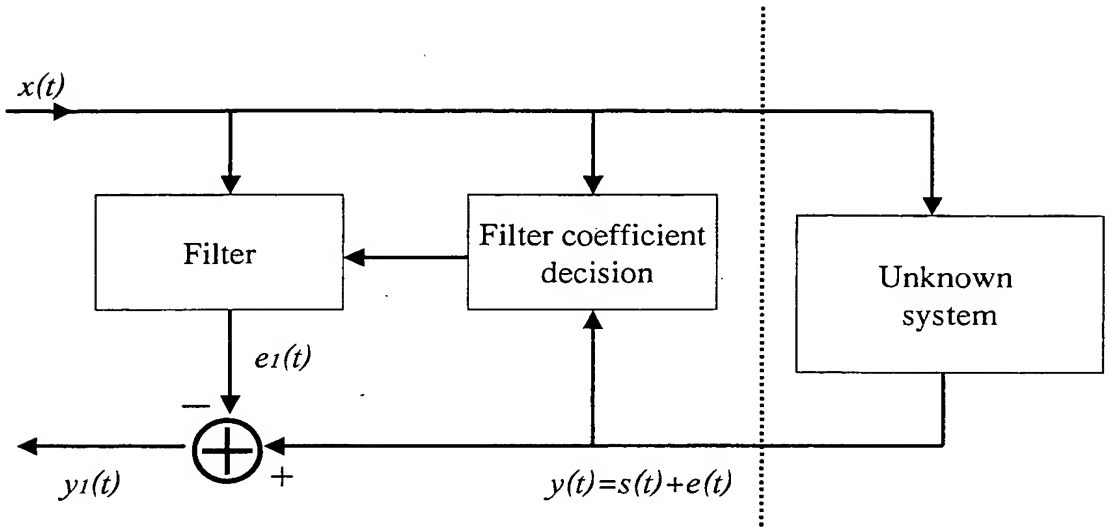
x(t)
Filter
Filter coefficient decision
Unknown system
e1(t)
−
+
y1(t)
y(t)=s(t)+e(t)

Figure 15

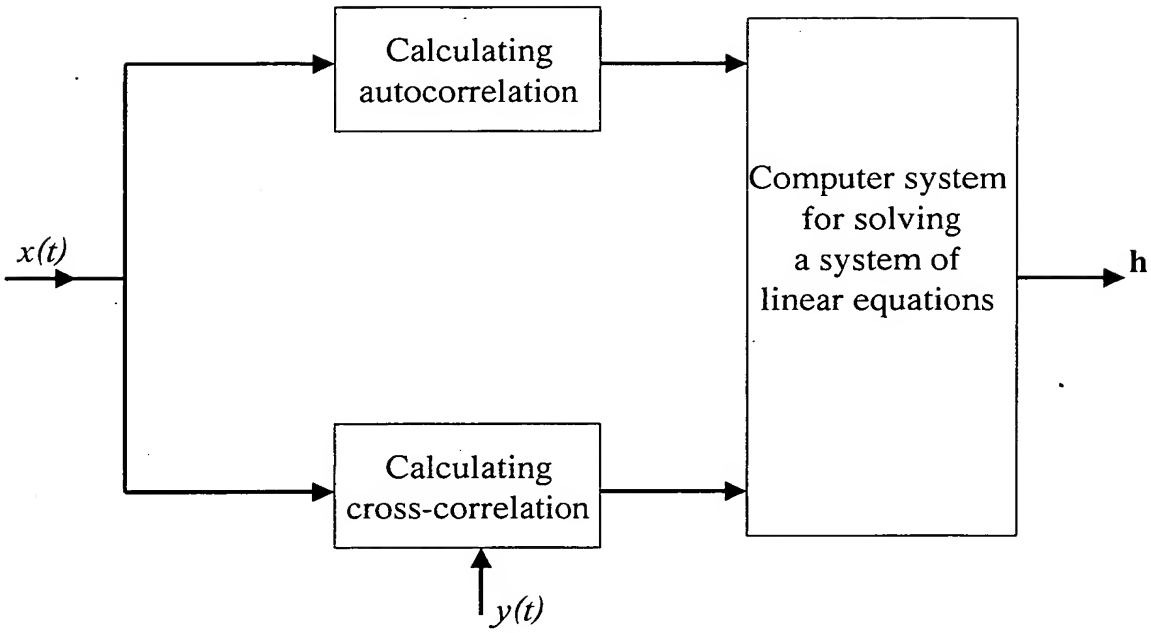
Calculating
autocorrelation
Computer system
for solving
a system of
linear equations
x(t)
h
Calculating
cross-correlation
y(t)

Figure 16